Sistemas Informáticos Curso 2006-2007

SISTEMA GESTOR DE BASES DE DATOS .NET

Presentado por:
Javier González Chacón, José María Marín del Valle e Irene Parrón Carrascal
Dirigido por:
Luis Hernández Yáñez

Facultad de Informática Universidad Complutense de Madrid

Abstract

The topic

The project deals with the research and development of an application which provides a tensile and handling tool for access to Data Bases. It has Windows and Web Interfaces. And it is independent of Data Bases which it is used.

It has been developed with Framework SDK 2,0 of .NET and Visual Studio.NET.

Why .NET?

Because .NET is the new bet of Microsoft for creating a new development platform of software which emphasize the transparency of networks, platform-independent and that allows rapid development of applications. In answer to the increasing market of the Web businesses, and able to compete to the Java platform of Sun Microsystems.

We thought that it was time to “enter .NETs world” and we began to learn its language, the C #, a great stranger for us although it involved an additional effort. Because the platform NET accepts several languages, (C #, VISUAL BASIC, C++, Nemerle, FORTRAN, Java, Python…etc), and it has enough capacity to accept practically any language.

C# is an Object Oriented language which was used to develop the . NET platform so it is the only one that allows to use all its potential. It is also certain that it is "a clean" language in the sense that it has been able to design it with more freedom and simplicity because it hasn’t been necessary to provide backward -compatibility.

In order to create applications for the platform .NET, both Web services and traditional applications (applications of console, applications of windows, services of Windows NT, etc.), Microsoft has published a kit of software development known as .NET Framework SDK, that includes the necessary tools for its development as its distribution and

execution. And Visual Studio .NET which permits the same but made through a visual interface based on windows. These tools, that we have commented previously, are what we have used to carry out the development of our application.

Key words: .NET, C#, ASP.net, Visual Studio.Net, Data Base, Microsoft

Resumen

Temática del proyecto.

El proyecto trata de la investigación y el desarrollo de una aplicación que proporcione una herramienta extensible y de fácil manejo para el acceso a bases de datos a través de interfaz Windows o Web. Y que sea independiente de la base de datos utilizada.

Ha sido íntegramente desarrollada con Framework SDK 2.0 de .NET y Visual Studio.NET.2005

¿Por qué .NET?

Porque .NET es la nueva apuesta multiplataforma de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes y que permita un rápido desarrollo de aplicaciones, en respuesta al creciente mercado de los negocios en entornos Web, como competencia a la plataforma Java de Sun Microsystems. Nos pareció interesante entrar en el "mundo .NET" y comenzar a aprender su lenguaje, C#, un gran desconocido para nosotros, a pesar del esfuerzo adicional que esto llevaba consigo.

Porque la plataforma .NET acepta varios lenguajes. Por ahora, C#, Visual Basic, C++ gestionado, Nemerle, FORTRAN, Java, Python, etc…, y tiene capacidad suficiente para aceptar prácticamente cualquier lenguaje.

Es C# el nuevo lenguaje orientado a objetos con el que se diseñó la plataforma .NET y es el único capaz de utilizar todo su potencial. También es cierto que es un lenguaje "limpio" en el sentido de que al no tener que proporcionar compatibilidad hacia atrás se ha tenido más libertad en el diseño y se ha puesto especial hincapié en la simplicidad.

Para crear aplicaciones para la plataforma .NET, tanto servicios Web como aplicaciones tradicionales (aplicaciones de consola, aplicaciones de ventanas, servicios de Windows NT, etc.), Microsoft ha publicado el denominado kit de desarrollo de software conocido como .NET Framework SDK, que incluye las herramientas necesarias tanto para su desarrollo como para su distribución y ejecución y Visual Studio.NET, que permite hacer todo la anterior desde una interfaz visual basada en ventanas. Estas herramientas, que ya hemos comentado anteriormente, son las que hemos empleado para llevar a cabo el desarrollo de nuestra aplicación.

Lista de palabras clave: .NET, C#, ASP.net, Visual Studio.Net, Base de Datos, Microsoft.

ÍNDICE DE CONTENIDOS

Descripción técnica del proceso del proyecto

1-.Entorno de trabajo

Microsoft.NET

Microsoft.NET es el conjunto de nuevas tecnologías en las que Microsoft ha estado trabajando durante los últimos años con el objetivo de obtener una multiplaforma sencilla y potente para distribuir el software en forma de servicios que puedan ser suministrados remotamente y que puedan comunicarse y combinarse unos con otros de manera totalmente independiente de la plataforma, lenguaje de programación y modelo de componentes con los que hayan sido desarrollados. Ésta es la llamada plataforma .NET, y a los servicios antes comentados se les denomina servicios Web.

Para crear aplicaciones para la plataforma .NET, tanto servicios Web como aplicaciones tradicionales, Microsoft ha publicado el denominado kit de desarrollo de software.NET Framework SDK, y Visual Studio.NET. Ambas herramientas pueden descargarse gratuitamente desde http://www.msdn.microsoft.com/net.

El concepto de Microsoft.NET también incluye al conjunto de nuevas aplicaciones que Microsoft y terceros han (o están) desarrollando para ser utilizadas en la plataforma .NET. Entre ellas podemos destacar aplicaciones desarrolladas por Microsoft tales como Windows.NET, Hailstorm, Visual Studio.NET, MSN.NET, Office.NET, y los nuevos servidores para empresas de Microsoft (SQL Server.NET, Exchange.NET, etc.)

Common Language Runtime (CLR)

Es el núcleo de la plataforma .NET. Es el motor encargado de gestionar la ejecución de las aplicaciones para ella desarrolladas y a las que ofrece numerosos servicios que

simplifican su desarrollo y favorecen su fiabilidad y seguridad. Las principales características y servicios que ofrece CLR son:

- Modelo de programación consistente: A todos los servicios y facilidades ofrecidos por CLR se accede de la misma forma: a través de un modelo de programación orientado a objetos. Esto es una diferencia importante respecto al modo de acceso a los servicios ofrecidos por algunos sistemas operativos actuales (por ejemplo, los de la familia Windows), en los que a algunos servicios se accede a través de llamadas a funciones globales definidas en DLLs y a otros a través de objetos (objetos COM en el caso de la familia Windows).

- Modelo de programación sencillo: Con CLR desaparecen muchos elementos complejos incluidos en los sistemas operativos actuales (registro de Windows, GUIDs, HRESULTS, IUnknown, etc.). CLR no es que abstraiga al programador de estos conceptos, sino que son conceptos que no existen en la plataforma .NET.

- Eliminación del "infierno de las DLLs": En la plataforma .NET desaparece el problema conocido como "infierno de las DLLs" que se da en los sistemas operativos actuales de la familia Windows, problema que consiste en que al sustituirse versiones viejas de DLLs compartidas por versiones nuevas puede que aplicaciones que fueron diseñadas para ser ejecutadas usando las viejas dejen de funcionar si las nuevas no son 100% compatibles con las anteriores. En la plataforma .NET las versiones nuevas de las DLLs pueden coexistir con las viejas, de modo que las aplicaciones diseñadas para ejecutarse usando las viejas podrán seguir usándolas tras la instalación de las nuevas. Esto, obviamente, simplifica mucho la instalación y desinstalación de software.

- Ejecución multiplataforma: CLR actúa como una máquina virtual, encargándose de ejecutar las aplicaciones diseñadas para la plataforma .NET. Es decir, cualquier plataforma para la que exista una versión del CLR podrá ejecutar cualquier aplicación

.NET. Microsoft ha desarrollado versiones del CLR para la mayoría de las versiones de Windows: Windows 95, Windows 98, Windows ME, Windows NT 4.0, Windows 2000, Windows XP y Windows CE (que puede ser usado en CPUs que no sean de la familia x86). Por otro lado Microsoft ha firmado un acuerdo con Corel para portar CLR a Linux y también hay terceros que están desarrollando de manera independiente versiones de libre distribución de CLR para Linux. Asimismo, dado que la arquitectura de CLR está totalmente abierta, es posible que en el futuro se diseñen versiones del mismo para otros sistemas operativos.

- Integración de lenguajes: Desde cualquier lenguaje para el que exista un compilador que genere código para la plataforma .NET es posible utilizar código generado para la misma usando cualquier otro lenguaje tal y como si de código escrito usando el primero se tratase. Microsoft ha desarrollado un compilador de C# que genera código de este tipo, así como versiones de sus compiladores de Visual Basic (Visual Basic.NET) y C++ (C++ con extensiones gestionadas) que también lo generan y una versión del intérprete de JScript (JScript.NET) que puede interpretarlo. La integración de lenguajes es tal que es posible escribir una clase en C# que herede de otra escrita en Visual Basic.NET que, a su vez, herede de otra escrita en C++ con extensiones gestionadas.

- Gestión de memoria: CLR incluye un recolector de basura que evita que el programador tenga que tener en cuenta cuándo ha de destruir los objetos que dejen de serle útiles. Este recolector es una aplicación que se activa cuando se quiere crear algún objeto nuevo y se detecta que no queda memoria libre para hacerlo, caso en que el recolector recorre la memoria dinámica asociada a la aplicación, detecta qué objetos hay en ella que no puedan ser accedidos por el código de la aplicación, y los elimina para limpiar la memoria de “objetos basura” y permitir la creación de otros nuevos. Gracias a este recolector se evitan errores de programación muy comunes como intentos de borrado de objetos ya borrados, agotamiento de memoria por olvido de eliminación de objetos inútiles o solicitud de acceso a miembros de objetos ya destruidos.

- Seguridad de tipos: CLR facilita la detección de errores de programación difíciles de localizar comprobando que toda conversión de tipos que se realice durante la ejecución de una aplicación .NET se haga de modo que los tipos origen y destino sean compatibles.

- Aislamiento de procesos: CLR asegura que desde código perteneciente a un determinado proceso no se pueda acceder a código o datos pertenecientes a otro, lo que evita errores de programación muy frecuentes e impide que unos procesos puedan atacar a otros. Esto se consigue gracias al sistema de seguridad de tipos antes comentado, pues evita que se pueda convertir un objeto a un tipo de mayor tamaño que el suyo propio, ya que al tratarlo como un objeto de mayor tamaño podría accederse a espacios en memoria ajenos a él que podrían pertenecer a otro proceso. También se consigue gracias a que no se permite acceder a posiciones arbitrarias de memoria.

- Tratamiento de excepciones: En CLR todos los errores que se puedan producir durante la ejecución de una aplicación se propagan de igual manera: mediante excepciones. Esto es muy diferente a como se venía haciendo en los sistemas Windows hasta la aparición de la plataforma .NET, donde ciertos errores se transmitían mediante códigos de error en formato Win32, otros mediante HRESULTs y otros mediante excepciones.

CLR permite que excepciones lanzadas desde código para .NET escrito en un cierto lenguaje se puedan capturar en código escrito usando otro lenguaje, e incluye mecanismos de depuración que pueden saltar desde código escrito para .NET en un determinado lenguaje a código escrito en cualquier otro. Por ejemplo, se puede recorrer la pila de llamadas de una excepción aunque ésta incluya métodos definidos en otros módulos usando otros lenguajes.

- Soporte multihilo: CLR es capaz de trabajar con aplicaciones divididas en múltiples hilos de ejecución que pueden ir evolucionando por separado en paralelo o intercalándose, según el número de procesadores de la máquina sobre la que se ejecuten.

Las aplicaciones pueden lanzar nuevos hilos, destruirlos, suspenderlos por un tiempo o hasta que les llegue una notificación, enviarles notificaciones, sincronizarlos, etc.

- Distribución transparente: CLR ofrece la infraestructura necesaria para crear objetos remotos y acceder a ellos de manera completamente transparente a su localización real, tal y como si se encontrasen en la máquina que los utiliza.

- Seguridad avanzada: CLR proporciona mecanismos para restringir la ejecución de ciertos códigos o los permisos asignados a los mismos según su procedendecia o el usuario que los ejecute. Es decir, puede no darse el mismo nivel de confianza a código procedente de Internet que a código instalado localmente o procedente de una red local; puede no darse los mismos permisos a código procedente de un determinado fabricante que a código de otro; y puede no darse los mismos permisos a un mismo código según el usuario que lo esté ejecutando o según el rol que éste desempeñe. Esto permite asegurar al administrador de un sistema que el código que se esté ejecutando no pueda poner en peligro la integridad de sus archivos, la del registro de Windows, etc.

- Interoperabilidad con código antiguo: CLR incorpora los mecanismos necesarios para poder acceder desde código escrito para la plataforma .NET a código escrito previamente a la aparición de la misma y, por tanto, no preparado para ser ejecutando dentro de ella. Estos mecanismos permiten tanto el acceso a objetos COM como el acceso a funciones sueltas de DLLs preexistentes (como la API Win32).

Como se puede deducir de las características comentadas, CLR lo que hace es gestionar la ejecución de las aplicaciones diseñadas para la plataforma .NET. Por esta razón, al código de estas aplicaciones se le suele llamar código gestionado, y al código no escrito para ser ejecutado directamente en la plataforma .NET se le suele llamar código no gestionado.

Microsoft Intermediate Language (MSIL)

Ninguno de los compiladores que generan código para la plataforma .NET produce código máquina para CPUs x86 ni para ningún otro tipo de CPU concreta, sino que genera código escrito en el lenguaje intermedio conocido como Microsoft Intermediate Lenguage (MSIL), CLR da a las aplicaciones la sensación de que se están ejecutando sobre una máquina virtual, y precisamente MSIL es el código máquina de esa máquina virtual. Es decir, MSIL es el único código que es capaz de interpretar CLR, y por tanto cuando se dice que un compilador genera código para la plataforma .NET lo que se está diciendo es que genera MSIL.

MSIL ha sido creado por Microsoft tras consultar a numerosos especialistas en la escritura de compiladores y lenguajes tanto del mundo académico como empresarial. Es un lenguaje de un nivel de abstracción mucho más alto que el de la mayoría de los códigos máquina de las CPUs existentes, e incluye instrucciones que permiten trabajar directamente con objetos (crearlos, destruirlos, inicializarlos, llamar a métodos virtuales, etcétera.), tablas y excepciones (lanzarlas, capturarlas y tratarlas).

Ya se comentó que el compilador de C# compila directamente el código fuente a MSIL, que Microsoft ha desarrollado nuevas versiones de sus lenguajes Visual Basic (Visual Basic.NET) y C++ (C++ con extensiones gestionadas) cuyos compiladores generan MSIL, y que ha desarrollado un intérprete de JScript (JScript.NET) que genera código MSIL. Pues bien, también hay numerosos terceros que han anunciado estar realizando versiones para la plataforma .NET de otros lenguajes como APL, CAML, Cobol, Eiffel, Fortran, Haskell, Java (J#), Mercury, ML, Mondrian, Oberon, Oz, Pascal, Perl, Python, RPG, Scheme y Smalltalk.

La principal ventaja del MSIL es que facilita la ejecución multiplataforma y la integración entre lenguajes al ser independiente de la CPU y proporcionar un formato común para el código máquina generado por todos los compiladores que generen código para .NET. Sin embargo, dado que las CPUs no pueden ejecutar directamente MSIL, antes de ejecutarlo

habrá que convertirlo al código nativo de la CPU sobre la que se vaya a ejecutar. De esto se encarga un componente CLR conocido como compilador JIT (Just-In-Time) o *jitter* que va convirtiendo dinámicamente el código MSIL a ejecutar en código nativo según sea necesario. Este jitter se distribuye en tres versiones:

- jitter normal: Es el que se suele usar por defecto y sólo compila el código MSIL a código nativo a medida que va siendo necesario, pues así se ahorra tiempo y memoria al evitarse tener que compilar innecesariamente código que nunca se ejecute. Para conseguir esto, el cargador de clases CLR sustituye inicialmente las llamadas a métodos de las nuevas clases que vaya cargando por llamadas a funciones auxiliares (stubs) que se encarguen de compilar el verdadero código del método. Una vez compilado, la llamada al stub es sustituida por una llamada directa al código ya compilado, con lo que posteriores llamadas al mismo no necesitarán compilación.

- jitter económico: Funciona de forma similar al jitter normal solo que no realiza ninguna optimización de código al compilar sino que traduce cada instrucción MSIL por su equivalente en el código máquina sobre la que se ejecute. Esta especialmente pensado para ser usado en dispositivos empotrados que dispongan de poca potencia de CPU y poca memoria, pues aunque genere código más ineficiente es menor el tiempo y memoria que necesita para compilar. Es más, para ahorrar memoria este jitter puede descargar código ya compilado que lleve cierto tiempo sin ejecutarse y sustituirlo de nuevo por el stub apropiado. Por estas razones, este es el jitter usado por defecto en Windows CE, sistema operativo que se suele incluir en los dispositivos empotrados antes mencionados.

Otra utilidad del jitter económico es que facilita la adaptación de la plataforma .NET a nuevos sistemas porque es mucho más sencillo de implementar que el normal. De este modo, gracias a él es posible desarrollar rápidamente una versión de CLR que pueda ejecutar aplicaciones gestionadas aunque sea de una forma poco eficiente, y una vez

desarrollada es posible centrarse en desarrollar el jitter normal para optimizar la ejecución de las mismas.

- prejitter: Se distribuye como una aplicación en línea de órdenes llamada ngen.exe mediante la que es posible compilar completamente cualquier ejecutable o biblioteca (cualquier ensamblado en general, aunque este concepto se verá más adelante) que contenga código gestionado y convertirlo a código nativo, de modo que posteriores ejecuciones del mismo se harán usando esta versión ya compilada y no se perderá tiempo en hacer la compilación dinámica.

La actuación de un jitter durante la ejecución de una aplicación gestionada puede dar la sensación de hacer que ésta se ejecute más lentamente debido a que ha de invertirse tiempo en las compilaciones dinámicas. Esto es cierto, pero hay que tener en cuenta que es una solución mucho más eficiente que la usada en otras plataformas como Java, ya que en .NET cada código no es interpretado cada vez que se ejecuta sino que sólo es compilado la primera vez que se llama al método al que pertenece. Es más, el hecho de que la compilación se realice dinámicamente permite que el jitter tenga acceso a mucha más información sobre la máquina en que se ejecutará la aplicación del que tendría cualquier compilador tradicional, con lo que puede optimizar el código para ella generado (por ejemplo, usando las instrucciones especiales del Pentium III si la máquina las admite, usando registros extra, incluyendo código *inline*, etc.) Además, como el recolector de basura de .NET mantiene siempre compactada la memoria dinámica las reservas de memoria se harán más rápido, sobre todo en aplicaciones que no agoten la memoria y, por tanto, no necesiten de una recolección de basura. Por estas razones, los ingenieros de Microsoft piensan que futuras versiones de sus jitters podrán incluso conseguir que el código gestionado se ejecute más rápido que el no gestionado.

Metadatos

En la plataforma .NET se distinguen dos tipos de módulos de código compilado: ejecutables (extensión .exe) y bibliotecas de enlace dinámico (extensión .dll generalmente) Ambos son ficheros que contienen definiciones de tipos de datos, y la diferencia entre ellos es que sólo los primeros disponen de un método especial que sirve de punto de entrada a partir del que es posible ejecutar el código que contienen haciendo una llamada desde la línea de comandos del sistema operativo. A ambos tipos de módulos se les suele llamar ejecutables portables (PE), ya que su código puede ejecutarse en cualquiera de los diferentes sistemas operativos de la familia Windows para los que existe alguna versión de CLR.

El contenido de un módulo no es sólo MSIL, sino que también consta de otras dos áreas muy importantes: la cabecera de CLR y los metadatos:

- La cabecera de CLR es un pequeño bloque de información que indica que se trata de un módulo gestionado e indica es la versión del CLR que necesita, cuál es su firma digital, cuál es su punto de entrada (si es un ejecutable), etc.

- Los metadatos son un conjunto de datos organizados en forma de tablas que almacenan información sobre los tipos definidos en el módulo, los miembros de éstos y sobre cuáles son los tipos externos al módulo a los que se les referencia en el módulo. Los metadatos de cada modulo los genera automáticamente el compilador al crearlo, y entre sus tablas se incluyen[1]:

Tabla	*Descripción*
ModuleDef	Define las características del módulo. Consta de un único elemento que almacena un identificador de versión de módulo (GUID creado por el compilador) y el nombre de fichero que se dio al módulo al compilarlo (así este nombre siempre estará disponible, aunque se renombre el fichero).
TypeDef	Define las características de los tipos definidos en el módulo. De cada tipo se almacena su nombre, su tipo padre, sus modificadores de acceso y referencias a los elementos de las tablas de miembros correspondientes a sus miembros.
MethodDef	Define las características de los métodos definidos en el módulo. De cada método se guarda su nombre, signatura (por cada parámetro se incluye una referencia al elemento apropiado en la tabla ParamDef), modificadores y posición del módulo donde comienza el código MSIL de su cuerpo.
ParamDef	Define las características de los parámetros definidos en el módulo. De cada parámetro se guarda su nombre y modificadores.
FieldDef	Define las características de los campos definidos en el módulo. De cada uno se almacena información sobre cuál es su nombre, tipo y modificadores.

PropertyDef	Define las características de las propiedades definidas en el módulo. De cada una se indica su nombre, tipo, modificadores y referencias a los elementos de la tabla MethodDef correspondientes a sus métodos set/get.
EventDef	Define las características de los eventos definidos en el módulo. De cada uno se indica su nombre, tipo, modificadores. y referencias a los elementos de la tabla MethodDef correspondientes a sus métodos add/remove.
AssemblyRef	Indica cuáles son los ensamblados externos a los que se referencia en el módulo. De cada uno se indica cuál es su nombre de fichero (sin extensión), versión, idioma y marca de clave pública.
ModuleRef	Indica cuáles son los otros módulos del mismo ensamblado a los que referencia el módulo. De cada uno se indica cuál es su nombre de fichero.
TypeRef	Indica cuáles son los tipos externos a los que se referencia en el módulo. De cada uno se indica cuál es su nombre y, según donde estén definidos, una referencia a la posición adecuada en la tabla AssemblyRef o en la tabla ModuleRef.
MemberRef	Indican cuáles son los miembros definidos externamente a los que se referencia en el módulo. Estos miembros pueden ser campos, métodos, propiedades o eventos; y

	de cada uno de ellos se almacena información sobre su nombre y signatura, así como una referencia a la posición de la tabla TypeRef donde se almacena información relativa al tipo del que es miembro.

Tabla 1: Principales tablas de metadatos.

Nótese que el significado de los metadatos es similar al de otras tecnologías previas a la plataforma .NET como lo son los ficheros IDL. Sin embargo, los metadatos tienen dos ventajas importantes sobre éstas: contiene más información y siempre se almacenan incrustados en el módulo al que describen, haciendo imposible la separación entre ambos. Además, como se verá más adelante, es posible tanto consultar los metadatos de cualquier módulo a través de las clases del espacio de nombres System.Reflection de la BCL como añadirles información adicional mediante atributos (se verá más adelante)

Ensamblados

Un ensamblado es una agrupación lógica de uno o más módulos o ficheros de recursos (ficheros .GIF, .HTML, etcétera.) que se engloban bajo un nombre común. Un programa puede acceder a información o código almacenados en un ensamblado sin tener que conocer cuál es el fichero en concreto donde se encuentran, por lo que los ensamblados nos permiten abstraernos de la ubicación física del código que ejecutemos o de los recursos que usemos. Por ejemplo, podemos incluir todos los tipos de una aplicación en un mismo ensamblado pero colocando los más frecuentemente usados en un cierto módulo y los menos usados en otro, de modo que sólo se descarguen de Internet los últimos si es que se van a usar.

Todo ensamblado contiene un manifiesto, que son metadatos con información sobre las características del ensamblado. Este manifiesto puede almacenarse en cualquiera de los módulos que formen el ensamblado o en uno específicamente creado para ello, siendo lo último necesario cuando sólo contiene recursos (ensamblado satélite).

Las principales tablas incluidas en los manifiestos son las siguientes:

Tabla	*Descripción*
AssemblyDef	Define las características del ensamblado. Consta de un único elemento que almacena el nombre del ensamblado sin extensión, versión, idioma, clave pública y tipo de algoritmo de dispersión usado para hallar los valores de dispersión de la tabla FileDef.
FileDef	Define cuáles son los archivos que forman el ensamblado. De cada uno se da su nombre y valor de dispersión. Nótese que sólo el módulo que contiene el manifiesto sabrá qué ficheros que forman el ensamblado, pero el resto de ficheros del mismo no sabrán si pertenecen o no a un ensamblado (no contienen metadatos que les indique si pertenecen a un ensamblado).
ManifestResourceDef	Define las características de los recursos incluidos en el módulo. De cada uno se indica

	su nombre y modificadores de acceso. Si es un recurso incrustado se indica dónde empieza dentro del PE que lo contiene, y si es un fichero independiente se indica cuál es el elemento de la tabla FileDef correspondiente a dicho fichero.
ExportedTypesDef	Indica cuáles son los tipos definidos en el ensamblado y accesibles desde fuera del mismo. Para ahorrar espacio sólo recogen los que no pertenezcan al módulo donde se incluye el manifiesto, y de cada uno se indica su nombre, la posición en la tabla FileDef del fichero donde se ha implementado y la posición en la tabla TypeDef correspondiente a su definición.
AssemblyProccesorDef	Indica en qué procesadores se puede ejecutar el ensamblado, lo que puede ser útil saberlo si el ensamblado contiene módulos con código nativo (podría hacerse usando C++ con extensiones gestionadas) Suele estar vacía, lo que indica que se puede ejecutar en cualquier procesador; pero si estuviese llena, cada elemento indicaría un tipo de procesador admitido según el formato de identificadores de procesador del fichero WinNT.h incluido con Visual Studio.NET (por ejemplo, 586 = Pentium, 2200 = Arquitectura IA64, etc.)

AssemblyOSDef	Indica bajo qué sistemas operativos se puede ejecutar el ensamblado, lo que puede ser útil si contiene módulos con tipos o métodos disponibles sólo en ciertos sistemas. Suele estar vacía, lo que indica que se puede ejecutar en cualquier procesador; pero si estuviese llena, indicaría el identificador de cada uno de los sistemas admitidos siguiendo el formato del WinNT.h de Visual Studio.NET (por ejemplo, 0 = familia Windows 9X, 1 = familia Windows NT, etc.) y el número de la versión del mismo a partir de la que se admite.

Tabla 2: Principales tablas de un manifiesto.

Para asegurar que no se haya alterado la información de ningún ensamblado se usa el criptosistema de clave pública RSA. Lo que se hace es calcular el código de dispersión SHA-1 del módulo que contenga el manifiesto e incluir tanto este valor cifrado con RSA (firma digital) como la clave pública necesaria para descifrarlo en algún lugar del módulo que se indicará en la cabecera de CLR. Cada vez que se vaya a cargar en memoria el ensamblado se calculará su valor de dispersión de nuevo y se comprobará que es igual al resultado de descifrar el original usando su clave pública. Si no fuese así se detectaría que se ha adulterado su contenido.

Para asegurar también que los contenidos del resto de ficheros que formen un ensamblado no hayan sido alterados lo que se hace es calcular el código de dispersión de éstos antes de cifrar el ensamblado y guardarlo en el elemento correspondiente a cada fichero en la tabla FileDef del manifiesto. El algoritmo de cifrado usado por defecto es

SHA-1, aunque en este caso también se da la posibilidad de usar MD5. En ambos casos, cada vez que se accede al fichero para acceder a un tipo o recurso se calculará de nuevo su valor de dispersión y se comprobará que coincida con el almacenado en FileDef.

Dado que las claves públicas son valores que ocupan muchos bytes (2048 bits), lo que se hace para evitar que los metadatos sean excesivamente grandes es no incluir en las referencias a ensamblados externos de la tabla AssemblyRef las claves públicas de dichos ensamblados, sino sólo los 64 últimos bits resultantes de aplicar un algoritmo de dispersión a dichas claves. A este valor recortado se le llama marca de clave pública.

Hay dos tipos de ensamblados: ensamblados privados y ensamblados compartidos. Los privados se almacenan en el mismo directorio que la aplicación que los usa y sólo puede usarlos ésta, mientras que los compartidos se almacenan en un caché de ensamblado global (GAC) y pueden usarlos cualquiera que haya sido compilada referenciándolos.

Los compartidos han de cifrase con RSA ya que lo que los identifica es en el GAC es su nombre (sin extensión) más su clave pública, lo que permite que en el GAC puedan instalarse varios ensamblados con el mismo nombre y diferentes claves públicas. Es decir, es como si la clave pública formase parte del nombre del ensamblado, razón por la que a los ensamblados así cifrados se les llama ensamblados de nombre fuerte. Esta política permite resolver los conflictos derivados de que se intente instalar en un mismo equipo varios ensamblados compartidos con el mismo nombre pero procedentes de distintas empresas, pues éstas tendrán distintas claves públicas.

También para evitar problemas, en el GAC se pueden mantener múltiples versiones de un mismo ensamblado. Así, si una aplicación fue compilada usando una cierta versión de un determinado ensamblado compartido, cuando se ejecute sólo podrá hacer uso de esa versión del ensamblado y no de alguna otra más moderna que se hubiese instalado en el GAC. De esta forma se soluciona el problema del infierno de las DLL comentado al principio del tema.

En realidad es posible modificar tanto las políticas de búsqueda de ensamblados (por ejemplo, para buscar ensamblados privados fuera del directorio de la aplicación) como la política de aceptación de ensamblados compartidos (por ejemplo, para que se haga automáticamente uso de las nuevas versiones que se instalen de DLLs compartidas) incluyendo en el directorio de instalación de la aplicación un fichero de configuración en formato XML con las nuevas reglas para las mismas. Este fichero ha de llamarse igual que el ejecutable de la aplicación pero ha de tener extensión .cfg.

Biblioteca de clase base (BCL)

La Biblioteca de Clase Base (BCL) es una biblioteca incluida en el *.NET Framework* formada por cientos de tipos de datos que permiten acceder a los servicios ofrecidos por CLR y a las funcionalidades más frecuentemente usadas a la hora de escribir programas. Además, a partir de estas clases prefabricadas el programador puede crear nuevas clases que mediante herencia extiendan su funcionalidad y se integren a la perfección con el resto de clases de la BCL. Por ejemplo, implementando ciertos interfaces podemos crear nuevos tipos de colecciones que serán tratadas exactamente igual que cualquiera de las colecciones incluidas en la BCL.

Esta biblioteca está escrita en MSIL, por lo que puede usarse desde cualquier lenguaje cuyo compilador genere MSIL. A través de las clases suministradas en ella es posible desarrollar cualquier tipo de aplicación, desde las tradicionales aplicaciones de ventanas, consola o servicio de Windows NT hasta los novedosos servicios Web y páginas ASP.NET. Es tal la riqueza de servicios que ofrece que incluso es posible crear lenguajes que carezcan de librería de clases propia y sólo se basen en la BCL -como C#.

Dada la amplitud de la BCL, ha sido necesario organizar las clases en ella incluida en espacios de nombres que agrupen clases con funcionalidades similares. Por ejemplo, los espacios de nombres más usados son:

Espacio de nombres	*Utilidad de los tipos de datos que contiene*
System	Tipos muy frecuentemente usados, como los los tipos básicos, tablas, excepciones, fechas, números aleatorios, recolector de basura, entrada/salida en consola, etc.
System.Collections	Colecciones de datos de uso común como pilas, colas, listas, diccionarios, etc.
System.Data	Manipulación de bases de datos. Forman la denominada arquitectura ADO.NET.
System.IO	Manipulación de ficheros y otros flujos de datos.
System.Net	Realización de comunicaciones en red.
System.Reflection	Acceso a los metadatos que acompañan a los módulos de código.
System.Runtime.Remoting	Acceso a objetos remotos.
System.Security	Acceso a la política de seguridad en

	que se basa CLR.
System.Threading	Manipulación de hilos.
System.Web.UI.WebControls	Creación de interfaces de usuario basadas en ventanas para aplicaciones Web.
System.Windows.Forms	Creación de interfaces de usuario basadas en ventanas para aplicaciones estándar.
System.XML	Acceso a datos en formato XML.

Tabla 3: Espacios de nombres de la BCL más usados.

Common Type System (CTS)

Common Type System (CTS) o Sistema de Tipo Común es el conjunto de reglas que han de seguir las definiciones de tipos de datos para que CLR las acepte. Es decir, aunque cada lenguaje gestionado disponga de su propia sintaxis para definir tipos de datos, en el MSIL resultante de la compilación de sus códigos fuente se han de cumplir las reglas del CTS. Algunos ejemplos de estas reglas son:

- Cada tipo de dato puede constar de cero o más miembros. Cada uno de estos miembros puede ser un campo, un método, una propiedad o un evento.

- No puede haber herencia múltiple, y todo tipo de dato ha de heredar directa o indirectamente de System.Object.

- Los modificadores de acceso admitidos son:

Modificador	*Código desde el que es accesible el miembro*
public	Cualquier código
private	Código del mismo tipo de dato
family	Código del mismo tipo de dato o de hijos de éste.
assembly	Código del mismo ensamblado
family and assembly	Código del mismo tipo o de hijos de éste ubicado en el mismo ensamblado
family or assembly	Código del mismo tipo o de hijos de éste, o código ubicado en el mismo ensamblado

Tabla 4: Modificadores de acceso a miembros admitidos por el CTS.

Common Language Specification (CLS)

Common Language Specification (CLS) o Especificación del Lenguaje Común es un conjunto de reglas que han de seguir las definiciones de tipos que se hagan usando un determinado lenguaje gestionado si se desea que sean accesibles desde cualquier otro lenguaje gestionado. Obviamente, sólo es necesario seguir estas reglas en las definiciones

de tipos y miembros que sean accesibles externamente, y no la en las de los privados. Además, si no importa la interoperabilidad entre lenguajes tampoco es necesario seguirlas. A continuación se listan algunas de reglas significativas del CLS:

- Los tipos de datos básicos admitidos son bool, char, byte, short, int, long, float, double, string y object Nótese pues que no todos los lenguajes tienen porqué admitir los tipos básicos enteros sin signo o el tipo decimal como lo hace C#.

- Las tablas han de tener una o más dimensiones, y el número de dimensiones de cada tabla ha de ser fijo. Además, han de indexarse empezando a contar desde 0.

- Se pueden definir tipos abstractos y tipos sellados. Los tipos sellados no pueden tener miembros abstractos.

- Las excepciones han de derivar de System.Exception, los delegados de System.Delegate, las enumeraciones de System.Enum, y los tipos por valor que no sean enumeraciones de System.ValueType.

- Los métodos de acceso a propiedades en que se traduzcan las definiciones get/set de éstas han de llamarse de la forma get_X y set_X respectivamente, donde X es el nombre de la propiedad; los de acceso a indizadores han de traducirse en métodos get_Item y set_Item; y en el caso de los eventos, sus definiciones add/remove han de traducirse en métodos add_X y remove_X.

- En las definiciones de atributos sólo pueden usarse enumeraciones o datos de los siguientes tipos: System.Type, string, char, bool, byte, short, int, long, float, double y object.

- En un mismo ámbito no se pueden definir varios identificadores cuyos nombres sólo difieran en la capitalización usada. De este modo se evitan problemas al acceder a ellos usando lenguajes no sensibles a mayúsculas.

Las enumeraciones no pueden implementar interfaces, y todos sus campos han de ser estáticos y del mismo tipo. El tipo de los campos de una enumeración sólo puede ser uno de estos cuatro tipos básicos: byte, short, int o long.

Origen y necesidad de un nuevo lenguaje

C# (leído en inglés “C Sharp” y en español “C Almohadilla”) es el nuevo lenguaje de propósito general diseñado por Microsoft para su plataforma .NET. Sus principales creadores son Scott Wiltamuth y Anders Hejlsberg, éste último también conocido por haber sido el diseñador del lenguaje Turbo Pascal y la herramienta RAD Delphi.

Aunque es posible escribir código para la plataforma .NET en muchos otros lenguajes, C# es el único que ha sido diseñado específicamente para ser utilizado en ella, por lo que programarla usando C# es mucho más sencillo e intuitivo que hacerlo con cualquiera de los otros lenguajes ya que C# carece de elementos heredados innecesarios en .NET. Por esta razón, se suele decir que C# es el lenguaje nativo de .NET.

La sintaxis y estructuración de C# es muy parecida a la de C++ o Java, puesto que la intención de Microsoft es facilitar la migración de códigos escritos en estos lenguajes a C# y facilitar su aprendizaje a los desarrolladores habituados a ellos. Sin embargo, su sencillez y el alto nivel de productividad son comparables con los de Visual Basic.

Un lenguaje que hubiese sido ideal utilizar para estos menesteres es Java, pero debido a problemas con la empresa creadora del mismo -Sun-, Microsoft ha tenido que desarrollar un nuevo lenguaje que añadiese a las ya probadas virtudes de Java las modificaciones que Microsoft tenía pensado añadirle para mejorarlo aún más y hacerlo un lenguaje orientado al desarrollo de componentes.

En resumen, C# es un lenguaje de programación que toma las mejores características de lenguajes preexistentes como Visual Basic, Java o C++ y las combina en uno solo. El hecho

de ser relativamente reciente no implica que sea inmaduro, pues Microsoft ha escrito la mayor parte de la BCL usándolo, por lo que su compilador es el más depurado y optimizado de los incluidos en el *.NET Framework SDK*.

Características de C#

Con la idea de que los programadores más experimentados puedan obtener una visión general del lenguaje, a continuación se recoge de manera resumida las principales características de C# Alguna de las características aquí señaladas no son exactamente propias del lenguaje sino de la plataforma .NET en general, y si aquí se comentan es porque tienen una repercusión directa en el lenguaje:

- Sencillez: C# elimina muchos elementos que otros lenguajes incluyen y que son innecesarios en .NET. Por ejemplo:
    - El código escrito en C# es autocontenido, lo que significa que no necesita de archivos adicionales al propio fuente tales como los de cabecera o IDL.
    - El tamaño de los tipos de datos básicos es fijo e independiente del compilador, sistema operativo o máquina para quienes se compile (no como en C++), lo que facilita la portabilidad del código.
    - No se incluyen elementos poco útiles de lenguajes como C++ tales como macros, herencia múltiple o la necesidad de un operador diferente del punto (.) acceder a miembros de espacios de nombres (::).
- Modernidad: C# incorpora en el propio lenguaje elementos que a lo largo de los años ha ido demostrándose son muy útiles para el desarrollo de aplicaciones y que en otros lenguajes como Java o C++ hay que simular, como un tipo básico decimal que permita realizar operaciones de alta precisión con reales de 128 bits (muy útil en el mundo financiero), la inclusión de una instrucción foreach que permita recorrer colecciones con

facilidad y es ampliable a tipos definidos por el usuario, la inclusión de un tipo básico string para representar cadenas o la distinción de un tipo bool específico para representar valores lógicos.

- Orientación a objetos: Como todo lenguaje de programación de propósito general actual, C# es un lenguaje orientado a objetos, aunque eso es más bien una característica del CTS que de C#. Una diferencia de este enfoque orientado a objetos respecto al de otros lenguajes como C++ es que el de C# es más puro en tanto que no admiten ni funciones ni variables globales sino que todo el código y datos han de definirse dentro de definiciones de tipos de datos, lo que reduce problemas por conflictos de nombres y facilita la legibilidad del código.

C# soporta todas las características propias del paradigma de programación orientada a objetos: encapsulación, herencia y polimorfismo.

En lo referente a la encapsulación es importante señalar que aparte de los típicos modificadores public, private y protected, C# añade un cuarto modificador llamado internal, que puede combinarse con protected e indica que al elemento a cuya definición precede sólo puede accederse desde su mismo ensamblado.

Respecto a la herencia a diferencia de C++ y al igual que Java- C# sólo admite herencia simple de clases ya que la múltiple provoca más quebraderos de cabeza que facilidades y en la mayoría de los casos su utilidad puede ser simulada con facilidad mediante herencia múltiple de interfaces. De todos modos, esto vuelve a ser más bien una característica propia de CTS que de C#.

Por otro lado y a diferencia de Java, en C# se ha optado por hacer que todos los métodos sean por defecto sellados y que los redefinibles hayan de marcarse con el modificador virtual (como en C++), lo que permite evitar errores derivados de redefiniciones accidentales. Además, un efecto secundario de esto es que las llamadas a los métodos serán más eficientes por defecto al no tenerse que buscar en la tabla de funciones

virtuales la implementación de los mismos a la que se ha de llamar. Otro efecto secundario es que permite que las llamadas a los métodos virtuales se puedan hacer más eficientemente al contribuir a que el tamaño de dicha tabla se reduzca.

- Orientación a componentes: La propia sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones más o menos complejas. Es decir, la sintaxis de C# permite definir cómodamente propiedades (similares a campos de acceso controlado), eventos (asociación controlada de funciones de respuesta a notificaciones) o atributos (información sobre un tipo o sus miembros).

- Gestión automática de memoria: Como ya se comentó, todo lenguaje de .NET tiene a su disposición el recolector de basura de CLR. Esto tiene el efecto en el lenguaje de que no es necesario incluir instrucciones de destrucción de objetos. Sin embargo, dado que la destrucción de los objetos a través del recolector de basura es indeterminista y sólo se realiza cuando éste se activa ─ya sea por falta de memoria, finalización de la aplicación o solicitud explícita en el fuente─, C# también proporciona un mecanismo de liberación de recursos determinista a través de la instrucción using.

- Seguridad de tipos: C# incluye mecanismos que permiten asegurar que los accesos a tipos de datos siempre se realicen correctamente, lo que permite evita que se produzcan errores difíciles de detectar por acceso a memoria no perteneciente a ningún objeto y es especialmente necesario en un entorno gestionado por un recolector de basura. Para ello se toman medidas del tipo:

    - Sólo se admiten conversiones entre tipos compatibles. Esto es, entre un tipo y antecesores suyos, entre tipos para los que explícitamente se haya definido un operador de conversión, y entre un tipo y un tipo hijo suyo del que un objeto del primero almacenase una referencia del segundo (*downcasting*) Obviamente, lo último sólo puede comprobarlo en tiempo de ejecución CLR y no el compilador,

por lo que en realidad CLR y el compilador colaboran para asegurar la corrección de las conversiones.

- No se pueden usar variables no inicializadas. El compilador da a los campos un valor por defecto consistente en ponerlos a cero y controla mediante análisis del flujo de control del fuente que no se lea ninguna variable local sin que se le haya asignado previamente algún valor.

- Se comprueba que todo acceso a los elementos de una tabla se realice con índices que se encuentren dentro del intervalo de la misma.

- Se puede controlar la producción de desbordamientos en operaciones aritméticas, informándose de ello con una excepción cuando ocurra. Sin embargo, para conseguirse un mayor rendimiento en la aritmética estas comprobaciones no se hacen por defecto al operar con variables sino sólo con constantes (se pueden detectar en tiempo de compilación).

- A diferencia de Java, C# incluye delegados, que son similares a los punteros a funciones de C++ pero siguen un enfoque orientado a objetos, pueden almacenar referencias a varios métodos simultáneamente, y se comprueba que los métodos a los que apunten tengan parámetros y valor de retorno del tipo indicado al definirlos.

- Pueden definirse métodos que admitan un número indefinido de parámetros de un cierto tipo, y a diferencia lenguajes como C/C++, en C# siempre se comprueba que los valores que se les pasen en cada llamada sean de los tipos apropiados.

- Instrucciones seguras: Para evitar errores muy comunes, en C# se han impuesto una serie de restricciones en el uso de las instrucciones de control más comunes. Por ejemplo, la guarda de toda condición ha de ser una expresión condicional y no aritmética, con lo que se evitan errores por confusión del operador de igualdad (==) con el de asignación (=); y todo caso de un switch ha de terminar en un break o goto que indique cuál es la

siguiente acción a realizar, lo que evita la ejecución accidental de casos y facilita su reordenación.

- Sistema de tipos unificado: A diferencia de C++, en C# todos los tipos de datos que se definan siempre derivarán, aunque sea de manera implícita, de una clase base común llamada System.Object, por lo que dispondrán de todos los miembros definidos en ésta clase (es decir, serán "objetos").

A diferencia de Java, en C# esto también es aplicable a los tipos de datos básicos. Además, para conseguir que ello no tenga una repercusión negativa en su nivel de rendimiento, se ha incluido un mecanismo transparente de *boxing* y *unboxing* con el que se consigue que sólo sean tratados como objetos cuando la situación lo requiera, y mientras tanto puede aplicárseles optimizaciones específicas.

El hecho de que todos los tipos del lenguaje deriven de una clase común facilita enormemente el diseño de colecciones genéricas que puedan almacenar objetos de cualquier tipo.

- Extensibilidad de tipos básicos: C# permite definir, a través de estructuras, tipos de datos para los que se apliquen las mismas optimizaciones que para los tipos de datos básicos. Es decir, que se puedan almacenar directamente en pila (luego su creación, destrucción y acceso serán más rápidos) y se asignen por valor y no por referencia. Para conseguir que lo último no tenga efectos negativos al pasar estructuras como parámetros de métodos, se da la posibilidad de pasar referencias a pila a través del modificador de parámetro ref.

- Extensibilidad de operadores: Para facilitar la legibilidad del código y conseguir que los nuevos tipos de datos básicos que se definan a través de las estructuras estén al mismo nivel que los básicos predefinidos en el lenguaje, al igual que C++ y a diferencia de Java, C# permite redefinir el significado de la mayoría de los operadores ─incluidos

los de conversión, tanto para conversiones implícitas como explícitas─ cuando se apliquen a diferentes tipos de objetos.

Las redefiniciones de operadores se hacen de manera inteligente, de modo que a partir de una única definición de los operadores ++ y -- el compilador puede deducir automáticamente como ejecutarlos de manera prefijas y postifja; y definiendo operadores simples (como +), el compilador deduce cómo aplicar su versión de asignación compuesta (+=) Además, para asegurar la consistencia, el compilador vigila que los operadores con opuesto siempre se redefinan por parejas (por ejemplo, si se redefine ==, también hay que redefinir !=).

También se da la posibilidad, a través del concepto de indizador, de redefinir el significado del operador [] para los tipos de dato definidos por el usuario, con lo que se consigue que se pueda acceder al mismo como si fuese una tabla. Esto es muy útil para trabajar con tipos que actúen como colecciones de objetos.

- Extensibilidad de modificadores: C# ofrece, a través del concepto de atributos, la posibilidad de añadir a los metadatos del módulo resultante de la compilación de cualquier fuente información adicional a la generada por el compilador que luego podrá ser consultada en tiempo ejecución a través de la librería de reflexión de .NET . Esto, que más bien es una característica propia de la plataforma .NET y no de C#, puede usarse como un mecanismo para definir nuevos modificadores.

- Versionable: C# incluye una política de versionado que permite crear nuevas versiones de tipos sin temor a que la introducción de nuevos miembros provoquen errores difíciles de detectar en tipos hijos previamente desarrollados y ya extendidos con miembros de igual nombre a los recién introducidos.

Si una clase introduce un nuevo método cuyas redefiniciones deban seguir la regla de llamar a la versión de su padre en algún punto de su código, difícilmente seguirían esta regla miembros de su misma signatura definidos en clases hijas previamente a la

definición del mismo en la clase padre; o si introduce un nuevo campo con el mismo nombre que algún método de una clase hija, la clase hija dejará de funcionar. Para evitar que esto ocurra, en C# se toman dos medidas:

- Se obliga a que toda redefinición deba incluir el modificador override, con lo que la versión de la clase hija nunca sería considerada como una redefinición de la versión de miembro en la clase padre ya que no incluiría override. Para evitar que por accidente un programador incluya este modificador, sólo se permite incluirlo en miembros que tengan la misma signatura que miembros marcados como redefinibles mediante el modificador virtual. Así además se evita el error tan frecuente en Java de creerse haber redefinido un miembro, pues si el miembro con override no existe en la clase padre se producirá un error de compilación.

- Si no se considera redefinición, entonces se considera que lo que se desea es ocultar el método de la clase padre, de modo que para la clase hija sea como si nunca hubiese existido. El compilador avisará de esta decisión a través de un mensaje de aviso que puede suprimirse incluyendo el modificador new en la definición del miembro en la clase hija para así indicarle explícitamente la intención de ocultación.

- Eficiente: En principio, en C# todo el código incluye numerosas restricciones para asegurar su seguridad y no permite el uso de punteros. Sin embargo, y a diferencia de Java, en C# es posible saltarse dichas restricciones manipulando objetos a través de punteros. Para ello basta marcar regiones de código como inseguras (modificador unsafe) y podrán usarse en ellas punteros de forma similar a cómo se hace en C++, lo que puede resultar vital para situaciones donde se necesite una eficiencia y velocidad procesamiento muy grandes.

- Compatible: Para facilitar la migración de programadores, C# no sólo mantiene una sintaxis muy similar a C, C++ o Java que permite incluir directamente en código escrito en C# fragmentos de código escrito en estos lenguajes, sino que CLR también ofrece, a

través de los llamados Platform Invocation Services (PInvoke), la posibilidad de acceder a código nativo escrito como funciones sueltas no orientadas a objetos tales como las DLLs de la API Win32. Nótese que la capacidad de usar punteros en código inseguro permite que se pueda acceder con facilidad a este tipo de funciones, ya que éstas muchas veces esperan recibir o devuelven punteros.

También es posible acceder desde código escrito en C# a objetos COM. Para facilitar esto, el *.NET Framework SDK* incluye una herramientas llamadas tlbimp y regasm mediante las que es posible generar automáticamente clases proxy que permitan, respectivamente, usar objetos COM desde .NET como si de objetos .NET se tratase y registrar objetos .NET para su uso desde COM.

Finalmente, también se da la posibilidad de usar controles ActiveX desde código .NET y viceversa. Para lo primero se utiliza la utilidad aximp, mientras que para lo segundo se usa la ya mencionada regasm.

2-.Idea inicial del proyecto

Nuestra aplicación es una herramienta extensible y de fácil manejo que proporciona acceso a diferentes tipos de bases de datos a través de interfaz Windows o Web. Una parte importante de nuestra aplicación radica en dotar, a un usuario sin conocimientos de informática, de la capacidad de mantener una o varias bases de datos. Entendiendo por mantener la capacidad de insertar, borrar, modificar y consultar los datos de las mismas.

Nuestra primera idea era generar y compilar código de la misma en tiempo de ejecución añadiendo lo como parte del programa para realizar una generación dinámica de los formularios y accesos a datos necesarios. Debido a las estimaciones de tiempo de que disponíamos y viendo que tratándose de la aplicación que teníamos entre manos la generación de código no iba a aportar demasiado realmente útil frente a la generación de formularios genéricos precompilados. Decidimos centrarnos en la aplicación y dejar de lado la parte de generación y compilación de código en tiempo de ejecución.

No obstante nos pareció que la generación de código aunque tediosa era un tema muy interesante tanto como para dedicar un proyecto paralelo a este tema, ya que es visible que la ingeniería del software esta evolucionando rápidamente, y cada vez se investiga más las formas de construir aplicaciones de manera automática Creando software que construye software, o generando reglas dinámicas que se obtendrán de bases de datos o de ficheros de configuración

3-.Planificación de tareas y temporal

Una vez fijados los objetivos y estimada la envergadura de nuestra aplicación comenzamos a planificar las tareas y el tiempo.

Identificamos tres partes bien diferenciadas:

- Interfaz Windows
- Interfaz Web
- Conexión con base de datos

Que nos permitían dividir el trabajo para ir avanzando en paralelo con la independencia que necesitábamos ya que los tres integrantes del equipo trabajamos y no era sencillo reunirnos para trabajar conjuntamente en el proyecto.

Con esta partición de tareas y manteniendo la comunicación formal vía mail, Messenger y reuniones periódicas, fuimos llevando a cabo el desarrollo progresivo de nuestra aplicación.

Las reuniones periódicas han sido una parte fundamental en la gestión de nuestro proyecto. En ellas hacíamos planificaciones a corto plazo hasta la siguiente reunión. Y exponíamos las dudas y/o problemas encontrados en cada una de las partes hasta ese momento de la misma. De esta manera íbamos solventándolos o tomando las decisiones que considerábamos adecuadas en función de de las estimaciones de complejidad, tiempo y recursos de que disponíamos.

4-.Procesos y actividades intermedias

1-.Familiarización con el lenguaje y entorno de trabajo

La primera barrera con la que nos encontramos fue el desconocimiento tanto del lenguaje con el que pensábamos desarrollar nuestra aplicación C#, como del las herramientas y entorno con el que íbamos a trabajar, Visual Studio 2005.

Conseguimos ir superando esa barrera poco a poco a base de una primera fase de búsqueda de documentación, mucha lectura y mucha práctica.

2-.Especificación de los requisitos de la aplicación

Definición de requisitos funcionales

Acciones:

- Conectar con base de datos

Descripción:

Permite conectarse a una base de datos concreta ya este definida en un servidor o en un fichero

Datos entrantes:

- Cadena de conexión
- Proveedor

Datos salientes:

- Nombres de las tablas que contiene la base de datos

- Alta

Descripción:

Permite al insertar un nuevo registro en una tabla de datos concreta de la base de datos.

Datos entrantes:

  - Nombre de la tabla
  - Registro a añadir

Datos salientes:

  - Número de filas dadas de alta

- Baja

Descripción:

Permite eliminar un registro en una tabla de datos concreta de la base de datos.

Datos entrantes:

  - Nombre de la tabla
  - Registro a eliminar

Datos salientes:

  - Número de filas eliminadas

- Modificación

Descripción:

Permite modificar los campos de un registro en una tabla de datos concreta de la base de datos.

Datos entrantes:

  - Nombre de la tabla
  - Registro a modificar

Datos salientes:

  - Número de filas modificadas

- Consulta simple

Descripción:

Permite consultar todos los registros de una tabla de datos concreta de la base de datos.

Datos entrantes:

  - Nombre de la tabla

Datos salientes:

  - Tabla seleccionada

- Consulta

Descripción:

Permite realizar consultas SQL sobre la base de datos

Datos entrantes:

- Consulta SQL

Datos salientes:

- Tabla resultado de la ejecución de la consulta

- Búsqueda

Descripción:

Permite realizar consultas sobre la base de datos de manera abstracta, sin indicar explícitamente la consulta SQL.

Datos entrantes:

- Campos a mostrar:

  Campos de cada tabla que deseo mostrar al ejecutar la consulta.

- Restricciones a la consulta:

  Restricciones con las que construyo la parte del *WHERE* de la consulta SQL a ejecutar.

- Tablas intermedias:

  Son las tablas intermedias que debo atravesar para ejecutar la consulta

Datos salientes:

- Tabla resultado de la ejecución de la consulta

- Búsqueda avanzada

Descripción:

Permite realizar consultas sobre la base de datos de manera abstracta, sin indicar explícitamente la consulta SQL. Requiere menos campos que la Búsqueda descrita anteriormente.

Datos entrantes:

- Campos a mostrar:

  Campos de cada tabla que deseo mostrar al ejecutar la consulta.

- Restricciones a la consulta:

  Restricciones con las que construyo la parte del *WHERE* de la consulta SQL a ejecutar.

Datos salientes:

- Tabla resultado de la ejecución de la consulta

Definición de requisitos de operativa

Operaciones:

- Conectar con base de datos

Descripción:

Permite al usuario conectarse a una base de datos concreta ya esté definida en un servidor o en un fichero.

Cada instancia de nuestro programa permitirá al usuario una única conexión simultánea.

Procedimiento:

1. El sistema le permite al usuario elegir entre:
    - Conectarse a una base de datos definida en un servidor

    Nota: En caso de que el usuario eligiera conectarse a una base de datos definida en un servidor deberá seleccionar además la base de datos de entre todas las que contenga el servidor.

    - Conectarse a una base de datos definida en un fichero
2. El usuario deberá introducir la cadena de conexión
3. El sistema permitirá al usuario seleccionar el proveedor deseado
    - Access
    - Oracle
    - SQLServer
4. El usuario solicitará la conexión

Condición posterior:

El sistema habrá ejecutado la acción:

- Conectar con base de datos

- Alta

Descripción:

Permite al usuario dar de alta un nuevo registro en una tabla concreta de la base de datos.

Procedimiento:

1. El sistema le permite al usuario elegir la tabla de la base de datos donde se insertará el nuevo registro.
2. El sistema presentará dinámicamente los campos a rellenar para efectuar el alta del registro.
3. El usuario rellenara los campos y solicitara la inserción.

Condición posterior:

El sistema habrá ejecutado la acción:

- Alta y mostrará la tabla con el nuevo registro ya insertado.

- Baja

Descripción:

Permite al usuario eliminar un registro en una tabla concreta de la base de datos.

Procedimiento:

1. El sistema le permite al usuario elegir la tabla de la base de datos de donde eliminará el registro.
2. El sistema presentará la tabla con todos sus registros.

3. El usuario seleccionará el registro a eliminar y solicitará la eliminación del mismo.

Condición posterior:

El sistema habrá ejecutado la acción:

- Baja y mostrará la tabla tras la eliminación efectuada.

- Modificación

Descripción:

Permite al usuario modificar un registro de una tabla concreta de la base de datos.

Procedimiento:

1. El sistema le permite al usuario elegir la tabla de la base de datos de donde modificará el registro.
2. El sistema presentará la tabla con todos sus registros.
3. El usuario seleccionará el registro a modificar y una modificado solicitará la actualización del mismo.

Condición posterior:

El sistema habrá ejecutado la acción:

- Modificación y mostrará la tabla tras la actualización efectuada.

- Consulta simple

Descripción:

Permite al usuario consultar todos los registros de una tabla de datos concreta de la base de datos.

Procedimiento:

1. El sistema le permite al usuario elegir la tabla de la base de datos que desea consultar

Condición posterior:

El sistema habrá ejecutado la acción:

- Consulta simple, mostrando la tabla solicitada.

- Consulta

Descripción:

Asiste al usuario de una manera sencilla a la hora de construir y ejecutar consultas SQL sobre la base de datos.

Aunque el usuario debe ser conocedor del lenguaje SQL.

Procedimiento:

1. Asistencia en SELECT:

   El sistema le permite al usuario seleccionar los campos de las tablas que estarán presentes en el resultado de la consulta.

2. Asistencia en WHERE:

   El sistema le permite al usuario seleccionar los campos de las tablas que con los que desea construir las restricciones. Y le facilita al máximo la construcción de las mismas de una manera asistida.

3. El usuario podrá modificar la consulta construida de manera manual antes de su ejecución.

4. El usuario solicitará la ejecución de la consulta construida.

Condición posterior:

El sistema habrá ejecutado la acción:

- Consulta, mostrando el resultado de la misma.

- Búsqueda

Descripción:

Asiste al usuario de una manera sencilla a la hora de construir y ejecutar consultas SQL sobre la base de datos.

Esta búsqueda está orientada a los usuarios desconocedores del lenguaje SQL pero que conocen las relaciones entre las tablas existentes en la base de datos.

Procedimiento:

1. Asistencia en SELECT:

   El sistema le permite al usuario seleccionar los campos de las tablas que estarán presentes en el resultado de la consulta.

2. Asistencia en WHERE:

   El sistema le permite al usuario seleccionar los campos de las tablas e incluir los valores de los mismos con los que desea construir las restricciones. Facilitándole al máximo la construcción de las mismas de una manera asistida.

   i. Asistencia en JOINT:

      El sistema permite al usuario incluir JOINT en sus cláusulas WHERE únicamente indicando las tablas intermedias que participan en él. Sin

necesidad de conocer como se enlazan ya que será el propio motor de base de datos el que las enlace a través de las claves ajenas de la base de datos.

3. El usuario solicitará la ejecución de la consulta construida.

Condición posterior:

El sistema habrá ejecutado la acción:

- Búsqueda, mostrando el resultado de la misma.

- Búsqueda avanzada

Descripción:

Asiste al usuario de una manera sencilla a la hora de construir y ejecutar consultas SQL sobre la base de datos.

Esta búsqueda está orientada a los usuarios desconocedores del lenguaje SQL y que tampoco conocen las relaciones entre las tablas existentes en la base de datos.

Procedimiento:

4. Asistencia en SELECT:

El sistema le permite al usuario seleccionar los campos de las tablas que estarán presentes en el resultado de la consulta.

5. Asistencia en WHERE:

El sistema le permite al usuario seleccionar los campos de las tablas e incluir los valores de los mismos con los que desea construir las restricciones. Facilitándole al máximo la construcción de las mismas de una manera asistida.

i. Asistencia en JOINT:

El sistema se encargará de encontrar las tablas intermedias y realizar los JOINTs pertinentes sin que el usuario deba preocuparse por ello.

6. El usuario solicitará la ejecución de la consulta construida.

Condición posterior:

El sistema habrá ejecutado la acción:

- Búsqueda avanzada, mostrando el resultado de la misma.

Definición de requisitos de administrabilidad

- *Idioma*: el idioma de las etiquetas de todos los formularios y pantallas programa será administrable. Pudiéndose escoger entre diferentes idiomas por ahora español e ingles.
- *Tipo de base de datos*: será administrable el tipo de base de datos accedida por la aplicación. Pudiéndose escoger entre SQLServer, Access y Oracle.
- *Interfaz de acceso*: también será administrable el tipo de interfaz que tendrá la aplicación pudiéndose elegir entre una interfaz Windows o una Web.

3-.Conexión con base de datos

Diseño de la conexión a base de datos:

Comenzamos realizando una clase de factorías para la creación de todos los objetos de bases de datos: conexión, adaptadores y comandos. Eso nos permitía aislarnos completamente de las bases de datos concretas y trabajar con Interfaces.

Una clase que realizaba el trabajo de bases de datos: altas, bajas modificaciones y consultas.

Y otra clase, que a través de la funcionalidad ofrecida por el método GetSchema() examinaba, la base de datos y nos permitía conocer por ejemplo: conocer las bases de datos que contenía un servidor, las tablas que contenía cada base de datos, las columnas contenidas en cada una de las tablas, las claves primarias y secundarias de cada tabla etcétera…

Teniendo este modelo a medio desarrollar comenzamos a hacer pruebas con distintas bases de datos y nos dimos cuenta de que la función GetSchema() de la conexión, que habíamos pensado genérica, tenía un comportamiento bien distinto al esperado.

Nosotros habíamos partido de la convicción de que el GetSchema() de la conexión nos suministraría igual información independientemente de la base de datos y fuera cual fuera el proveedor de acceso. Tras varias pruebas con diferentes bases de datos y proveedores nos dimos cuenta de que no solo no aceptaba siempre los mismos valores en sus parámetros sino que además no devolvía la misma información en unos proveedores .NET que en otros. Ni siquiera en el caso de acceder a la misma Base de Datos desde dos proveedores distintos.

Toda esta problemática provocó que tuviéramos que replantearnos el diseño inicial de la aplicación y modificarlo, viéndose afectado el horizonte temporal de nuestro proyecto.

Finalmente el diseño quedo de la siguiente manera:

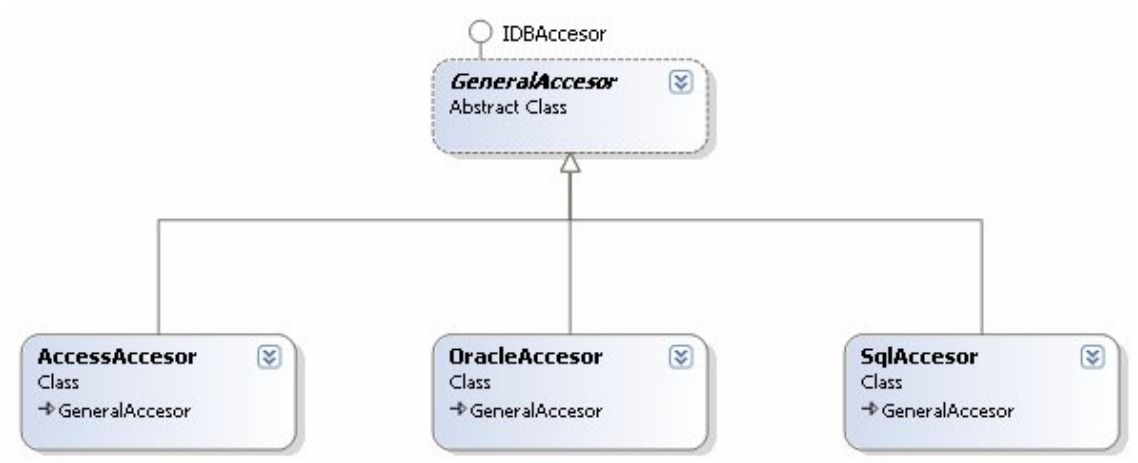
Una Interfaz IDBAccesor que se encarga de suministrarle toda la funcionalidad necesaria a las interfaces de nuestra aplicación.

Una clase abstracta que se encarga de implementa la interfaz GeneralAccesor y es clase padre de las clases de cada Base de Datos.

Y varias clases de accesores XXXAccesor que implementan la funcionalidad específica de cada Base de Datos y que no implementa GeneralAccesor clase de la cual heredan.

Una vez realizada la refactorización del código debería ser el GeneralAccesor quien implementara todo el trabajo de altas, bajas, modificaciones y consultas a modo genérico independientemente de la base de datos. Y dejar el trabajo de examinar la base de datos a los XXXAccesores. Pero aún falta un poco de refactorización para conseguir que el desarrollo para una nueva base de datos sea todavía más sencillo. Consistiría en hacer que las funciones de generación de comandos estuvieran implementadas en el GeneralAccesor.

En el gráfico que mostramos a continuación se muestran las relaciones entre las clases descritas:



Problemática encontrado en el desarrollo del la conexión a base de datos:

- *Claves foráneas (FKs):*

Para la exploración de la base de datos se utilizaba el método GetSchema() del objeto conexión. La información que nos devolvía era incompleta ya que no nos decía a que columna y tabla hacía referencia la clave foránea.

El nuevo diseño de clases nos permite realizar una exploración más exhaustiva y específica de la base de datos concreta. Aunque seguimos usando GetSchema() para muchas de las consultas donde la información que nos devuelve nos es suficiente.

- *El método GetSchema():*

Aun con el diseño adoptado, hemos pretendido que el código necesario para acceder a una nueva base de datos fuera el menos posible, maximizando así la opción de poner en el cuerpo del GeneralAccesor la mayoría del mismo.

Pero esto tampoco ha sido sencillo debido a la funcionalidad que nos ofrecía el método GetSchema(). Que recibe diferentes claves de búsqueda y restricciones. Y devuelve distintos datos siempre dependientes de la bases de datos accedida y del proveedor utilizado.

- *Claves foráneas en Microsoft Access:*

Otro de los problemas al que nos hemos enfrentado y que más tiempo de investigación nos ha consumido ha sido el cómo encontrar las claves foráneas en las tablas de Access. Y aunque la solución resultó ser simple parecía esconderse de nosotros.

La solución se basa en lo siguiente, el proveedor OleDB con el que accedemos a Access tiene en la propia conexión una función que el resto de conexiones no tenían las demás conexiones. Se trata de GetOleDbSchemaTable y esta función nos devolvía la información que estábamos buscando.

- Búsquedas problemática.

La tercera problemática, que todavía no hemos logrado solucionar, es como vincular las tablas de las queries. Esta problemática aunque solucionable no es tan fácil como en un principio pensamos. En un primer momento pensamos seguir las claves ajenas en un sentido u otro para unir todas las tablas. Pero no fue hasta que tuvimos una tabla que relacionaba a otras dos cuando nos dimos cuenta que podría ser que no todas las claves ajenas fueran en el mismo sentido. Y que para unir todas las tablas podría ser necesaria una tabla que únicamente tuviera claves ajenas salientes hacia otras tablas. Fue en este momento en el que descubrimos la autentica dificultad del algoritmo, ya casi al límite del plazo de entrega.

4-.Interfaz Windows

Tras la instalación del NET Framework SDK, y Visual Studio.NET. Comenzamos a hacer pequeñas pruebas para familiarizarnos con el entorno de trabajo y nos pusimos a diseñar y desarrollar la interfaz Windows de la aplicación.

La interfaz consta de una serie de formularios, para los cuales hemos utilizado un desarrollo multi-idioma, que nos permite adaptar fácilmente nuestra aplicación a distintos idiomas.

Los formularios son los siguientes:

- *Asistente de conexión:*

Como su propio nombre indica este formulario nos asiste a la hora de realizar la conexión a la base de datos deseada ya esté definida en un fichero o en un servidor

- *Selección de la BBDD:*

Este formulario nos permite escoger la base de datos deseada, en el caso de esta esté definida en un servidor.

- *Selección de tabla:*

Este formulario nos permite escoger la tabla de la base de datos sobre la que vamos a trabajar.

- *MDIParent:*

Las siglas MDI significan Multiple Document Interface Aplication.

El formulario MDI padre es un tanto especial. Es un formulario padre que contiene una serie de formularios MDI hijos, subventanas del formulario padre, con las que el usuario interactúa.

En nuestra aplicación este formulario padre nos permite acceder a una tabla concreta de la base de datos para realizar sobre ella una serie de operaciones: altas, bajas modificaciones, consultas simples, consultas, búsquedas y búsquedas avanzadas.

Y los formularios hijos son todos aquellos que se van creando dentro del formulario padre al solicitar cada una de las operaciones descritas en el párrafo anterior.

- *MDI hijos:*
    - *Alta:*

      Este formulario nos permite dar de alta nuevos registros en la tabla seleccionada.

    - *Baja*

      Este formulario nos eliminar registros de la tabla seleccionada.

    - *Modificación*

      Este formulario nos permite modificar registros en la tabla seleccionada.

    - *Consulta Simple*

      Este formulario nos permite consultar todos los registros de la tabla seleccionada.

    - *Consulta*

      Este formulario nos permite construir y ejecutar consultas SQL sobre la tabla seleccionada. De manera asistida aunque conociendo el lenguaje SQL.

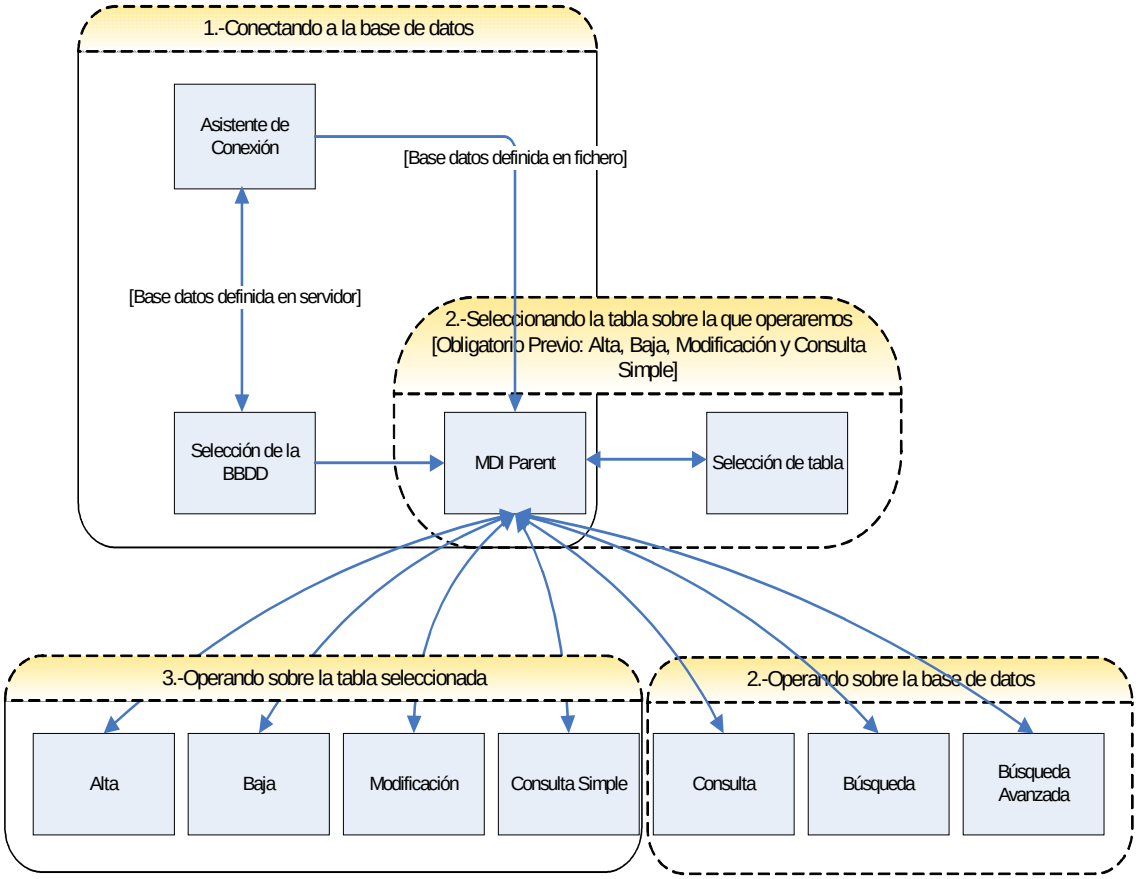
    - *Búsqueda*

      Este formulario nos permite construir y ejecutar consultas SQL sobre la tabla seleccionada. De manera asistida y sin necesidad de conocer el lenguaje SQL, aunque si las relaciones entre las tablas de la base de datos.

- *Búsqueda avanzada*

  Este formulario nos permite construir y ejecutar consultas SQL sobre la tabla seleccionada. De manera asistida y sin necesidad de conocer el lenguaje SQL, ni las relaciones entre las tablas de la base de datos.

La relación entre los formularios descritos se muestra en el diagrama que mostramos a continuación:



Algunos de los formularios descritos tienen características especiales. Este es el caso de los de Alta y Modificación.

En estos formularios los campos que en el caso del alta se usan para rellenar los datos del nuevo registro y en el de la modificación para modificarlos son auto generables, dependen completamente de la tabla sobre la que este operando en ese momento.

Otra característica especial de los formularios comentados es que disponen de un control lupa que les permite conectar con las claves foráneas de la tabla seleccionada. Para el desarrollo de este control hemos tenido que aprender a desarrollar dlls de control con C#.

5-.Desarrollo de la Interfaz Web

Tras la instalación del correspondiente servidor de aplicaciones IIS (Internet Information server).

Nos pusimos a familiarizarnos y a realizar pequeñas pruebas con la tecnología ASP.net con la que hemos desarrollado la Interfaz Web.

ASP.net

Es la tecnología de desarrollo de aplicaciones Web comercializado por la plataforma .NET de Microsoft, sucesora de la tecnología Active Server Pages (ASP).

Frente a las antiguas ASPs:

- Incorpora un modelo declarativo a la programación web:

Los controles de servidor funcionan en una página Web simplemente declarándolos. Cuando se carga la página ASP.NET, se instancian los controles listados en la página ASP y es responsabilidad del control emitir código HTML que el navegador pueda entender.

- Impone un cierto orden sobre el modelo de programación estándar ASP.

- Separa la porción basada en Script de una página Web de su contenido.

- Posee controles adicionales incluidos en el FrameWork. Una Herramienta para la construcción de reportes, con medios automáticos para exportarlos a XLS o PDF, y CristalReport. Además permite separar completamente la Interfaz de la lógica del Negocio permitiendo mayores niveles de abstracción en el desarrollo de aplicaciones Web.

- Y facilita el empleo de AJAX gracias al ToolKit de ASP.Net Ajax.

Una vez familiarizados con ASP.net comenzamos con el diseño de nuestra interfaz Web. Diseñando nuestra Interfaz. Esta compuesta por una serie de paginas que nombraremos a continuación, con las ofrecemos una operativa similar a la de la interfaz Windows y para las cuales hemos utilizado un desarrollo multi-idioma, que nos permite adaptar fácilmente nuestra aplicación a distintos idiomas.

Las páginas son las siguientes:

- *Asistente de conexión:*

Como su propio nombre indica esta página nos asiste a la hora de realizar la conexión a la base de datos deseada ya esté definida en un fichero o en un servidor

- *Selección de la BBDD:*

Esta página nos permite escoger la base de datos deseada, en el caso de esta esté definida en un servidor.

- *Selección de tabla y operación:*

Esta página nos permite escoger la tabla de la base de datos sobre la que vamos a trabajar y la operación que vamos a realizar sobre ella. O simplemente escoger la operación que vamos a realizar sobre la base de datos sin escoger ninguna tabla en concreto, para el caso de la consulta y las búsquedas.

- *Alta:*

Esta página nos permite dar de alta nuevos registros en la tabla seleccionada.

- *Baja*

Esta página nos eliminar registros de la tabla seleccionada.

- *Modificación*

Esta página nos permite modificar registros en la tabla seleccionada.

- *Consulta Simple*

Esta página nos permite consultar todos los registros de la tabla seleccionada.

- *Consulta*

Esta página nos permite construir y ejecutar consultas SQL sobre la tabla seleccionada. De manera asistida aunque conociendo el lenguaje SQL.

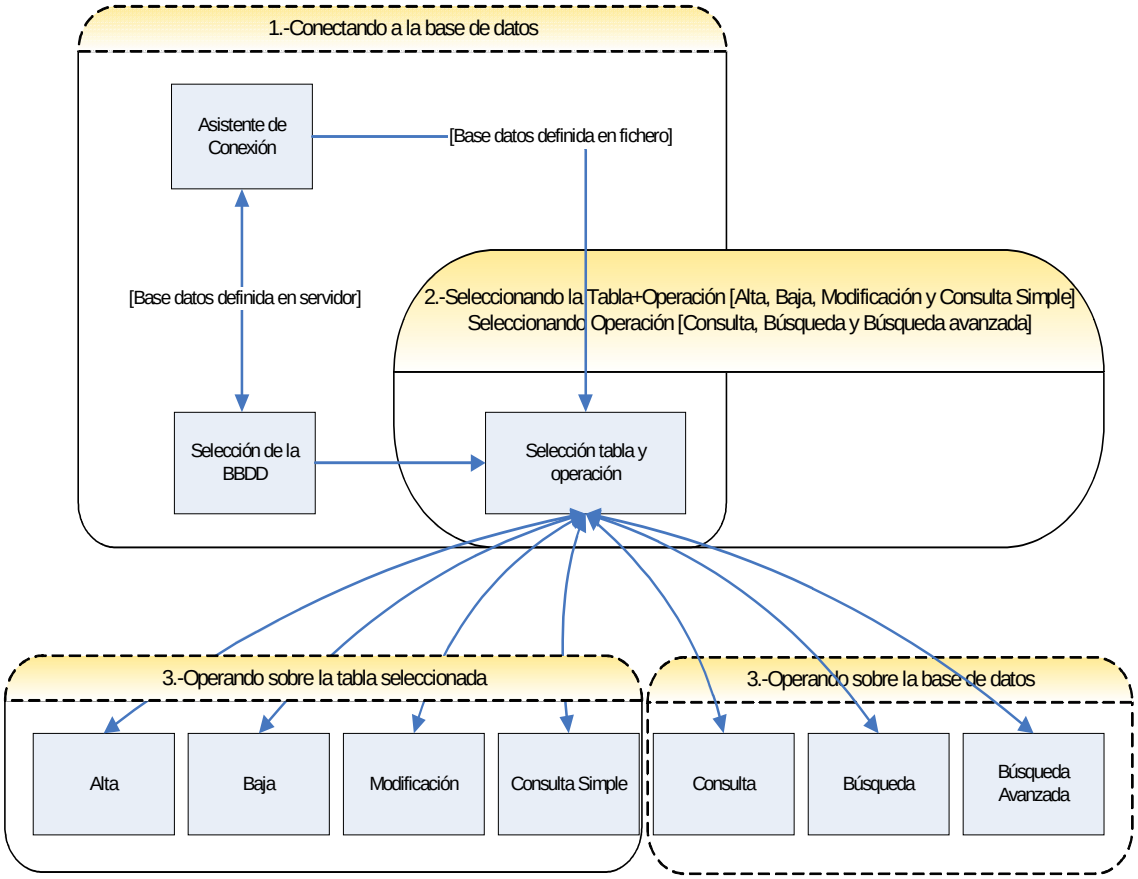
- *Búsqueda*

Esta página nos permite construir y ejecutar consultas SQL sobre la tabla seleccionada. De manera asistida y sin necesidad de conocer el lenguaje SQL, aunque si las relaciones entre las tablas de la base de datos.

- *Búsqueda avanzada*

Esta página nos permite construir y ejecutar consultas SQL sobre la tabla seleccionada. De manera asistida y sin necesidad de conocer el lenguaje SQL, ni las relaciones entre las tablas de la base de datos.

La relación entre las páginas descritas se muestra en el diagrama que mostramos a continuación:



A medida que avanzábamos en el diseño e íbamos documentándonos más nos fuimos dando cuenta que las tendencias actuales de desarrollo Web apuntan hacia el uso de AJAX (Asynchronous JavaScript And XML), que es una técnica de desarrollo Web para crear aplicaciones interactivas.

Éstas se ejecutan en el cliente, es decir, en el navegador de los usuarios, y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad en la misma.

Inicialmente nos planteamos el hacer uso de estas tecnologías en nuestro desarrollo pero la tediosa depuración de los javascripts que modelaban las respuestas del servidor junto al trabajo de investigación y pruebas que era necesario. Nuevamente mermó notablemente los

plazos temporales y esto nos llevo a replanificarlo como una posible mejora o ampliación de nuestra aplicación.

5-.Resultado final

Es una aplicación orientada sobre todo a usuarios desconocedores del lenguaje SQL, a los cuales proporciona una herramienta multilingüe, extensible, y de fácil manejo para acceder y operar con distintos tipos de bases de datos a través de interfaz Windows o Web.

Ha sido íntegramente desarrollada con Framework SDK 2.0 de .NET y Visual Studio.NET. 2005 Y en principio esta desarrollada para SQL Server 2005 y Access 2003 teniendo a medio desarrollo el acceso a Oracle debido a la falta de una BBDD para poder realizar las pruebas necesarias.

De todos modos hay que reconocer que nos hubiera gustado llegar a una refactorización mayor del código para conseguir que el desarrollo para incluir un nuevo tipo de base de datos sea aun si cabe más sencillo. Esta refactorización consistiría en hacer que las funciones de generación de comandos estuvieran implementadas en el GeneralAccesor la clase abstracta, padre de las clases de cada Base de Datos, que se encarga de implementa la interfaz GeneralAccesor de nuestra conexión con base de datos.

6-.Aplicación práctica

Esta aplicación está destinada a solucionar la necesidad de acceso a bases de datos de usuarios sin conocimientos informáticos. Les permite acceder a diferentes bases de datos: Access, SQLServer, Oracle etcétera y sin demasiado desarrollo adicional es fácilmente extensible a cualquier tipo de base de datos. Además para una mayor viabilidad el programa esta provisto de una funcionalidad multilenguaje lo que ayudaría a su distribución no solo dentro del territorio español sino también en el extranjero simplemente creando un nuevo fichero de recursos con el lenguaje deseado.

Posibles usos:

- Generación de aplicaciones genéricas para PYMEs, únicamente seria necesario desarrollar y configurar adecuadamente la base de datos concreta.

  Para este fin se debería crear un buen diseño de la base de datos de stock, facturación y contabilidad y obtener un generador de informes manejable para el usuario final. Y de esta manera añadiendo fotos a los productos y con las búsquedas actuales y las que estén en desarrollo se obtendría un producto útil y de bajo coste debido a su multiplicidad de usos.

Por ejemplo imaginemos que tenemos un supuesto cliente que nos dice:

“Necesito un programa con el que pueda acceder a mis distintas bases de datos porque tengo datos en un SQLServer de un programa que destino a facturación y tengo otros datos de un programa que me hizo un amigo que están en Access. Además tengo sucursales de mi negocio en Bilbao y Cataluña por lo que me gustaría que idioma de la aplicación fuera administrable.”

Otro ejemplo podría ser:

"Necesito un programa para mi negocio con el que pueda acceder las distintas bases de datos que manejo que son muy heterogéneas. Y además debido al constante crecimiento con respecto al número de bases de datos diferentes, quiero tener la certeza de que será extensible en tiempo record y con poco desarrollo adicional, en caso de incluir una nueva."

Otro posible uso podría ser:

- Generación de una aplicación orientada a la pre-venta.

  Habitualmente es difícil y caro (tanto en cuestión de tiempo como de recursos) mostrar una demo del producto final al cliente con la que pueda decidir la viabilidad del proyecto y su retorno de la inversión. Es conocido también que la aplicación además de útil ha de tener una interfaz amigable, este punto también podría solucionarse con esta orientación del uso de nuestro motor de bases de datos. Para este planteamiento seria importante desarrollar una interfaz útil y amigable que le pueda entrar por los ojos al cliente, esto unido a la posibilidad de mostrar la aplicación con los datos obtenidos de su propia base de datos y en su propia lengua creemos que seria una gran baza para la obtención del proyecto.

Conclusiones

La independencia de bases de datos en .NET

Después de haber estado trabajando sobre el tema hemos podido comprobar que a pesar de que las bases de datos proveen mucha cantidad de información sobre sí mismas. Y en la mayoría de los casos dicha información es común aún tratándose de diferentes tipos de bases de datos. La manera de llegar hasta cada una de ellas es todavía muy distinta.

Nos hemos ido dando cuenta de que aunque aparentemente cualquier tipo de base de datos tenía los mismos métodos de acceso a su información, dichos métodos trabajaban sobre argumentos diferentes en cada caso y devolviendo diferentes tipos de resultados.

Por tanto y a pesar de que los proveedores de bases de datos en .Net nos presentan una interfaz común para acceder a dicha información. Hemos podido comprobar que aún es imposible obtener una abstracción total respecto al tipo de base de datos utilizada. Sin llegar a realizar algo de desarrollo especifico para solventar las pequeñas diferencias en las maneras específicas de acceder a la información de cada una de ellas.

Las similitudes entre J2EE y .NET

- El propósito tanto de J2EE como de la plataforma .NET es facilitar y simplificar el desarrollo de aplicaciones empresariales o corporativas de cara al comercio electrónico. Las JSP (Java Server Pages) son muy similares a ASP .Net (Active Server Pages) y los EJB (Enterprise JavaBeans) son muy similares a los COM/COM+ de Microsoft.

- Los servidores de aplicaciones J2EE y .Net proporcionan un modelo de acceso de componentes a datos y de lógica del negocio, separados por una capa intermedia de presentación implementada mediante ASP .Net (:Net) ó Servlets (J2EE).

- Visual Basic .Net y C# son lenguajes orientados a objetos, al igual que Java, y en el diseño de todos ha tenido mucha importancia la existencia de Internet.

- Tanto J2EE como .Net proporcionan las herramientas necesarias para crear Servicios Web.

- Ambos J2EE y .Net han sido diseñados para soportar múltiples plataformas.

Comparando J2EE y .NET

Ventajas de .NET frente a J2EE:

- Una ventaja importante del entorno .Net frente a J2EE es la posibilidad de emplear múltiples lenguajes de programación, mientras que J2EE sólo trabaja con uno: Java. Esta diversidad de lenguajes es obligatoria por la misma variedad de problemas distintos a resolver. Ejemplo: un lenguaje moderno y orientado a objetos como Java puede resultar ineficaz a la hora de abordar problemas que involucren cálculos matemáticos masivos y complejos, mientras que esos mismos cálculos pueden ser abordados mucho más adecuadamente con un lenguaje tan primitivo como Fortran. Por otro esta característica multilenguaje posibilita así que programadores de terceros lenguajes pasen a esta plataforma reduciendo el tiempo de aprendizaje y entrenamiento.

- Las herramientas de desarrollo incluidas por Microsoft en Visual Studio .Net son mucho más simples, intuitivas y sencillas de manejar que las herramientas de desarrollo equivalentes en J2EE.

- C# es un lenguaje interesante, una rama evolutiva más del árbol de los lenguajes orientados a objetos.

- Microsoft ha impulsado los servicios Web y ha resaltado su importancia. Por ello la plataforma .Net se ha diseñado considerando los servicios Web siendo estos servicios

propios de la plataforma. Ofrece una nueva versión de ASP, ASP .Net, que puede considerarse un entorno de programación "de verdad" en lugar de un entorno basado en scripts. En términos de Microsoft ".Net fue construido para la integración a través de los servicios Web XML usando protocolos y formatos de ficheros como SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language), y UDDI (Universal Description, Discovery, and Integration)". Comparativamente, .Net va por delante de J2EE con respecto a servicios Web y estos servicios son propios de la plataforma. Y aunque J2EE respondió con el lanzamiento del Java Web Services Developer Pack, la facilidad, rapidez y sencillez con la que se pueden construir servicios Web con el Asistente de servicios Web de Visual Studio .Net, es muy superior a la de las herramientas para construir servicios Web dentro del entorno de J2EE.

Ventajas de J2EE frente a NET:

- Las implementaciones de J2EE pueden adquirirse a distintas compañías, mientras que .Net solo puede comprarse a Microsoft. El hecho de que haya distintas organizaciones implementando J2EE ofrece mayor variedad para los usuarios finales y permite la existencia de una cierta competencia entre ellas para obtener mejores productos que no existe en el caso de Microsoft y .Net.

- Debido al proceso evolutivo de los productos de Microsoft, y en muchos casos, por motivos de compatibilidad la seguridad frente a virus informáticos de los productos de Microsoft es menor que los basados en Java, pues desde un comienzo Java se fundamentó en un estricto modelo de seguridad.

- Las aplicaciones Java pueden correr en una amplia gama de sistemas operativos (desde sistemas empresariales como Windows 2000, OS/390, Solaris, HP-UX, IRIX u otras versiones de Unix hasta en sistemas orientados más a ordenadores personales como Mac OS, Windows 9x ó Linux, y en sistemas operativos para dispositivos móviles) y de arquitecturas hardware. Hasta la fecha, .Net corre solamente sobre sistemas operativos de

Microsoft (aunque esta situación podría cambiar en el futuro), siendo J2EE el único entorno de desarrollo que ofrece una independencia real de la plataforma.

- La tecnología Java es una tecnología abierta (en el sentido de que el código de la plataforma completa puede ser obtenido, revisado y estudiado por cualquiera que esté interesado). Esto posibilita que los desarrolladores puedan conocer y entender completamente cómo hace las cosas Java y aprovecharlo para sus aplicaciones. En contraposición, solo el código fuente del Nuevo lenguaje C# de la plataforma .Net ha sido abierto al público general (aunque Microsoft permite a compañías con las que le unen intereses comunes el acceso al código fuente de ciertas partes de .Net).

- Aunque Java fue creado originalmente por una compañía: Sun MicroSystems, lo cierto es que J2EE es ahora el producto de la colaboración de más de 400 empresas y organizaciones de todo tipo (público, privado sin ánimo de lucro, privado con ánimo de lucro, y de normalización en ámbitos nacionales e internaciones). La plataforma .Net es el producto de una sola compañía, que aunque haya implementado algunos estándares en .Net y esté intentando conseguir que ciertas tecnologías se conviertan en estándares "oficiales", no puede tener el mismo consenso que .J2EE (sobretodo teniendo en cuenta que la mayor parte de su código no es público).

- La tecnología Java goza ya de una cierta veteranía frente a .NET que le ha permitido probar su eficacia en muchos entornos y situaciones empresariales distintas.

El futuro

Hasta el momento, la plataforma J2EE es la única plataforma que corre en múltiples sistemas operativos y múltiples máquinas hardware y cuyos usuarios pueden seleccionar la implementación que más le convenga. Esta plataforma corre actualmente no solo en ordenadores doméstico o servidores o estaciones de trabajo, sino también en multitud de dispositivos como teléfonos móviles, agendas electrónicas, componentes electrónicos

industriales de automatización, etc. Su implantación en el mercado de la tos teléfonos móviles GMS y UMTS puede asegurar un próspero futuro, puesto que es más que probable que acaben convirtiéndose en un medio generalizado de acceso a Internet.

Tal y como se señaló antes, es perfectamente posible que Microsoft proporcione en el futuro entornos de ejecución .NET para diferentes plataformas de forma similar a Sun (como para Windows 64 bits sobre Itanium ó para un Windows CE sobre un Pocket PC), pero aún no lo ha hecho para plataformas no Windows, y sería raro que lo hiciera por su propia estrategia de mercado hasta la fecha. Aún así, llegaría con un retraso considerable con respecto a J2EE.

Sin embargo los servicios Web aún deben recorrer un largo camino entre estándares, proveedores de servicios y consumidores de los mismos, pero hasta el momento Microsoft y específicamente .Net llevan una cierta ventaja sobre J2EE. Aún así, sigue planeando la cuestión de la independencia de la plataforma. Si .Net no acaba siendo realmente multiplataforma, posiblemente J2EE predominará en el desarrollo de servicios Web para grandes empresas y para servicios Web clientes que requieran no depender de una plataforma específica, y .Net predominará en el desarrollo de servicios Web para pequeñas y medianas empresas que usan Windows.

Posibles extensiones del proyecto

Ampliar los lenguajes soportados:

Creando nuevos ficheros de recursos ".resx" con el formato de nombre adecuado al idioma que se desee incluir y las mismas entradas que el resto de los ficheros de idiomas ya definidos.

La aplicación cogerá el idioma del thread que se esté ejecutando y si no existe fichero para el mismo, cogerá el de por defecto.

Ampliar las bases de datos soportadas:

Creando una nueva clase que herede de la clase abstracta GeneralAccesor e implemente todos los métodos de la interfaz IDBAccesor.

Generación de código:

Generar y compilar código de la propia aplicación en tiempo de ejecución añadiendo lo como parte del programa para realizar una generación dinámica de los formularios y accesos a datos necesarios. Aunque este tipo de aplicación seria un desarrollo paralelo a nuestro proyecto que no aportaría muchas diferencias a los formularios precompilados. Pues proporciona dos soluciones distintas para resolver el mismo problema.

No obstante la generación de código es un tema interesante ya que cada vez está tomando más fuerza la búsqueda de nuevas formas de construir aplicaciones de manera automática, creando software que construye software. Y hay cada vez más herramientas que prueba de que la ingeniería del software esta evolucionando rápidamente.

Ejemplos de algunas de estas herramientas de generación de código son las siguientes:

Genexus, es una herramienta inteligente para la creación de aplicaciones en diferentes lenguajes de programación:

http://www.genexus.com/portal/hgxpp001.aspx

Codecharge otra herramienta para la generación de código:

http://www.yessoftware.com/index2.php

Mpo:

http://javahispano.net/projects/mpo/

Hacer uso de AJAX en la interfaz Web:

Durante el desarrollo de nuestra Interfaz fuimos viendo que las tendencias actuales de desarrollo Web apuntan hacia el uso de AJAX.

¿Qué es Ajax?

Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente, es decir, en el navegador de los usuarios, y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad de la misma.

AJAX es una combinación de tres tecnologías ya existentes:

- XHTML (o HTML) y hojas de estilos en cascada (CSS) para el diseño que acompaña a la información.

- Document Object Model (DOM) accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones ECMAScript como JavaScript y JScript, para mostrar e interactuar dinámicamente con la información presentada.

- El objeto XMLHttpRequest para intercambiar datos asincrónicamente con el servidor web. En algunos frameworks y en algunas situaciones concretas, se usa un objeto iframe en lugar del XMLHttpRequest para realizar dichos intercambios.

- XML es el formato usado comúnmente para la transferencia de vuelta al servidor, aunque cualquier formato puede funcionar, incluyendo HTML preformateado, texto plano, JSON y hasta EBML.

Bibliografía

Para familiarizarnos con el lenguaje C# hay algunos manuales que nos han resultado de gran ayuda y que por ello recomendamos:

Páginas:

http://programacion.com/tutorial.php?id=csharp.

http://www.ajlopez.net/ItemVe.php?Id=2076

Libros:

C# al descubierto / Joseph Mayo

Professional C#, 2nd ed. / Simon Robinsons

The C# programming language / Anders Hejlsberg, Scott Wiltamuth, Peter Golde

Para el desarrollo de la interfaz Windows nos ha sido de gran ayuda la consultar:

Páginas:

http://msdn2.microsoft.com/es-es/library/default.aspx

Libros:

Windows forms programming in C# / Chris Sells

Para el familiarizarnos con ASP.net y desarrollar de la interfaz Web recomendamos consultar:

Paginas:

http://www.willydev.net/Descargas/Cursos/AspNet/

http://www.es-asp.net/tutoriales-asp-net/tutorial-61-97.aspx

http://asp.net/

http://desarrolloweb.com/

http://es.gotdotnet.com/quickstart/aspplus/

Libros:

Build your own ASP.NET 2.0 web site using C# & VB : [the ultimate ASP.NET begginner's guide] / by Cristian Darie and Zak Ruvalcaba

Sobre el significado y uso de Ajax recomendamos algunas páginas:

http://www.librosweb.es/ajax/

http://www.ajaxpro.info/

http://www.codeproject.com/Ajax/AJAX.asp

http://ajax.schwarz-interactive.de/csharpsample/default.aspx

Libros:

Professional Ajax / Nicholas C. Zakas, Jeremy McPeak, Joe Fawcett

Anexo

Manual de usuario de la aplicación

Instalación

1. Entrar en la carpeta InterfazWindows\Proyecto2007\Release
2. Hacer doble clic en: “setup.exe”
3. Seguir las indicaciones del asistente de instalación.
4. Finalizada la instalación de la aplicación, que por defecto se instalará en “C:\Archivos de programa\UCM\Proyecto2007\”, hacer doble clic en “InterfazWindows.exe”

Interfaz Windows

Una vez instalado el programa no aparecerá el formulario principal del mismo:

En el que debemos realizar lo siguiente:

1. Elegir entre:

   - Conectar a una base de datos definida en un servidor.

   - O conectar a una base de datos definida en un fichero.

2. Introducir la cadena de conexión
3. Y seleccionar el proveedor deseado en la lista desplegable.
4. Aceptar o cancelar la operación

Si cancelamos la operación nos saldremos de la aplicación.

Si aceptamos existen dos posibilidades:

- En caso de haber seleccionado conectar a una base de datos definida en un servidor, tendremos que pasar por este formulario previo, antes de llegar al que comentaremos después.

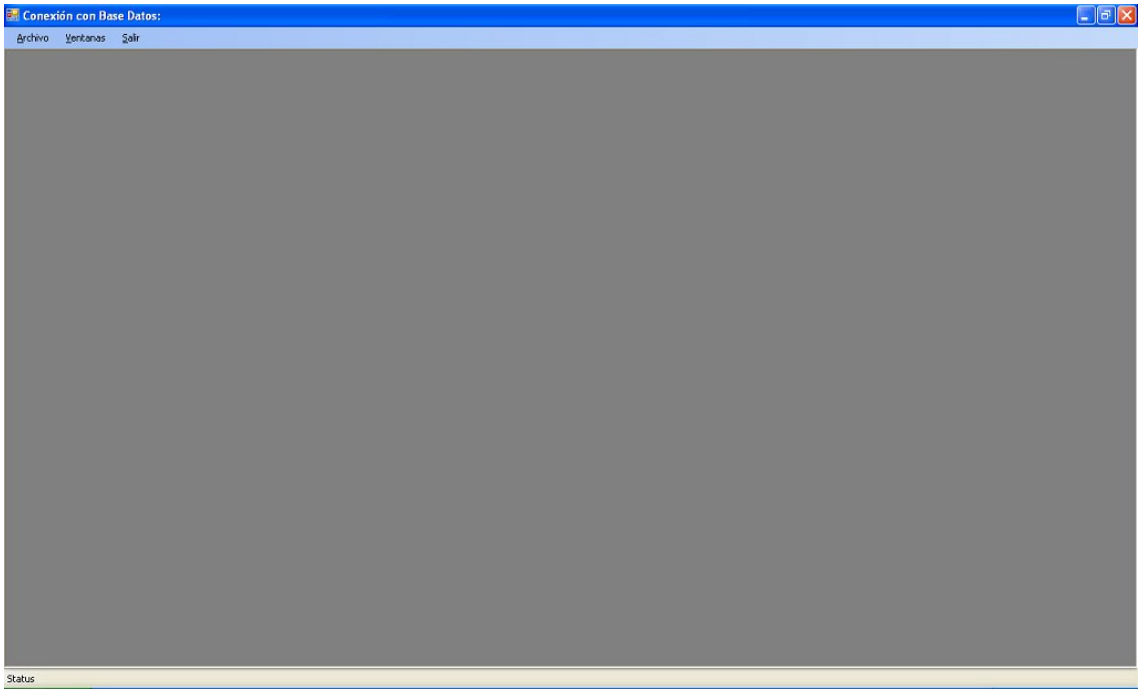
Pasos a seguir:

1. Seleccionar la base de datos concreta con la que queremos conectar, de entre las disponibles en el servidor.
2. Aceptar o cancelar la operación

Si cancelamos la operación pasamos al formulario anterior, formulario principal del programa.

Si aceptamos la operación pasamos a la página que comentamos a continuación.

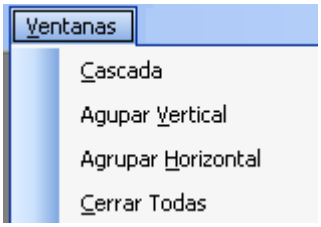
- En caso de haber seleccionado conectar a una base de datos definida en un fichero, pasaremos directamente (sin pasar por la página previa) al formulario que mostramos a continuación:



Desde este formulario padre iremos abriendo los formularios hijos con los que operaremos sobre la base de datos concreta. Podremos tener varias ventanas (formularios hijos) abiertas simultáneamente y será el propio padre el encargado de controlar que no se abran hijos repetidos.

Solo se permitirán abrir varias altas, bajas modificaciones y consultas siempre que sean sobre tablas diferentes de la base de datos. Y en caso de solicitar una operación repetida la aplicación nos remitirá a la ventana que ya teníamos abierta.

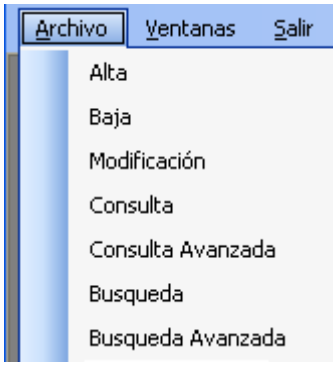
Las ventanas podrán agruparse desde el menú ventanas tal y como indicamos en la imagen que se muestra a continuación:



Solicitud de operaciones, pasos a seguir:

Si lo que deseamos hacer es realizar un: alta, una baja, una modificación o una consulta sobre una tabla concreta de la base de datos.

1. Seleccionamos la operación deseada, en el menú Archivo del formulario anterior



2. Y una vez seleccionada la operación se nos presentará una pantalla donde deberemos escoger la tabla de la base de datos sobre la que vamos a operar.

3. Tenemos las opciones siguientes:

   - Aceptar, de manera que se nos presentará el formulario correspondiente con la operación y la tabla seleccionada.

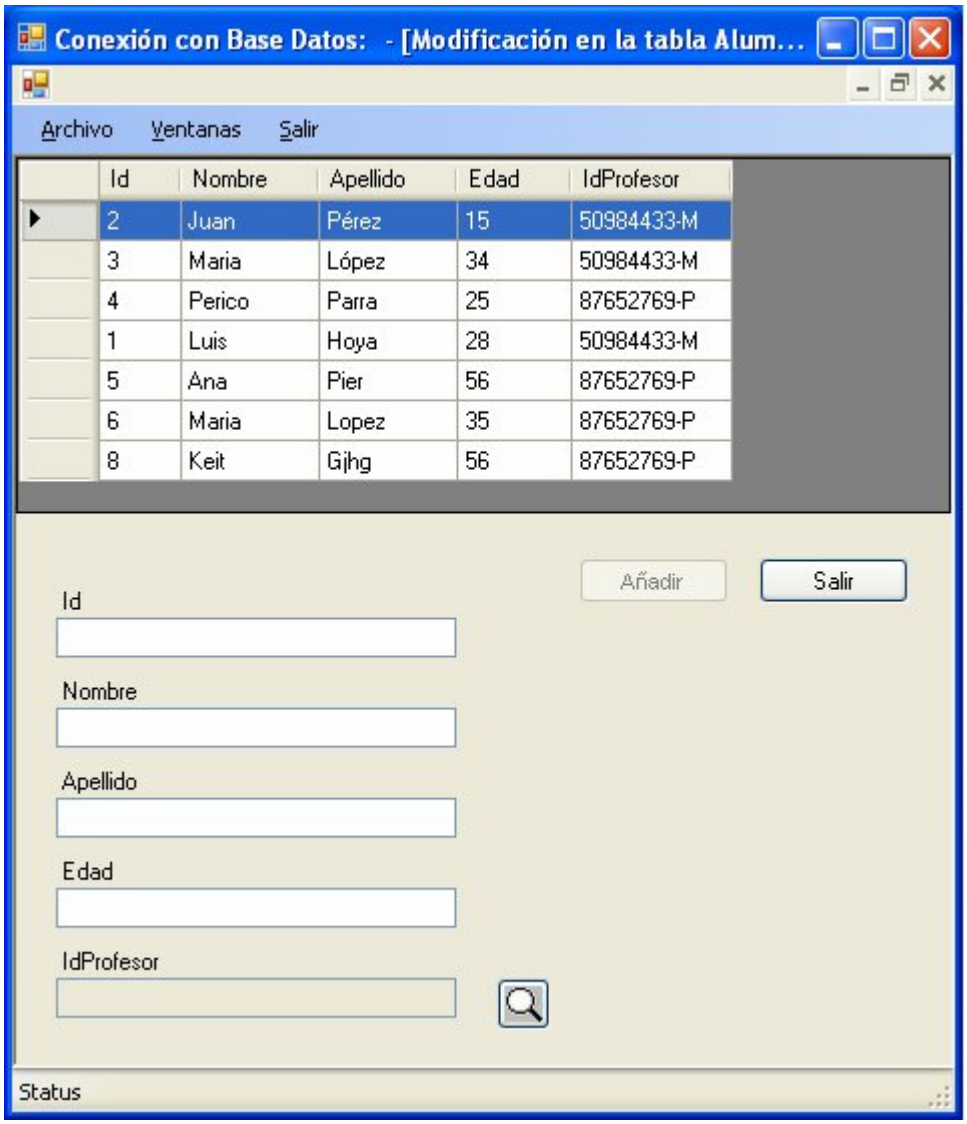
   - Cancelar, que nos cerrará el formulario de selección de tabla y permitiéndosenos volver a elegir una operación en el menú archivo del formulario de partida.

En cambio si deseamos hacer una: consulta avanzada, una búsqueda o una búsqueda avanzada sobre la base de datos

1. Únicamente seleccionaremos en el menú archivo del formulario, la operación que queremos realizar sobre la base de datos.

A continuación explicaremos cada una de las ventanas desde las que operamos sobre la base de datos:

- Alta:



Pasos a seguir:

1. Se rellenan los todos los campos necesarios para el alta del nuevo registro.

   Nota: Los campos clave ajena no serán editables manualmente, sino a través de un control lupa que nos permitirá seleccionar la fila referenciada y se traerá a nuestro formulario origen la clave ajena correspondiente. Ver campo IdProfesor de la imagen anterior.

2. Se pulsa el botón "Añadir" para solicitar la inserción del registro en la tabla concreta y se actualizan automáticamente los datos de la misma que aparecen en la parte superior del formulario.

Nota: con el botón "Salir" cerramos el formulario.

- Baja:

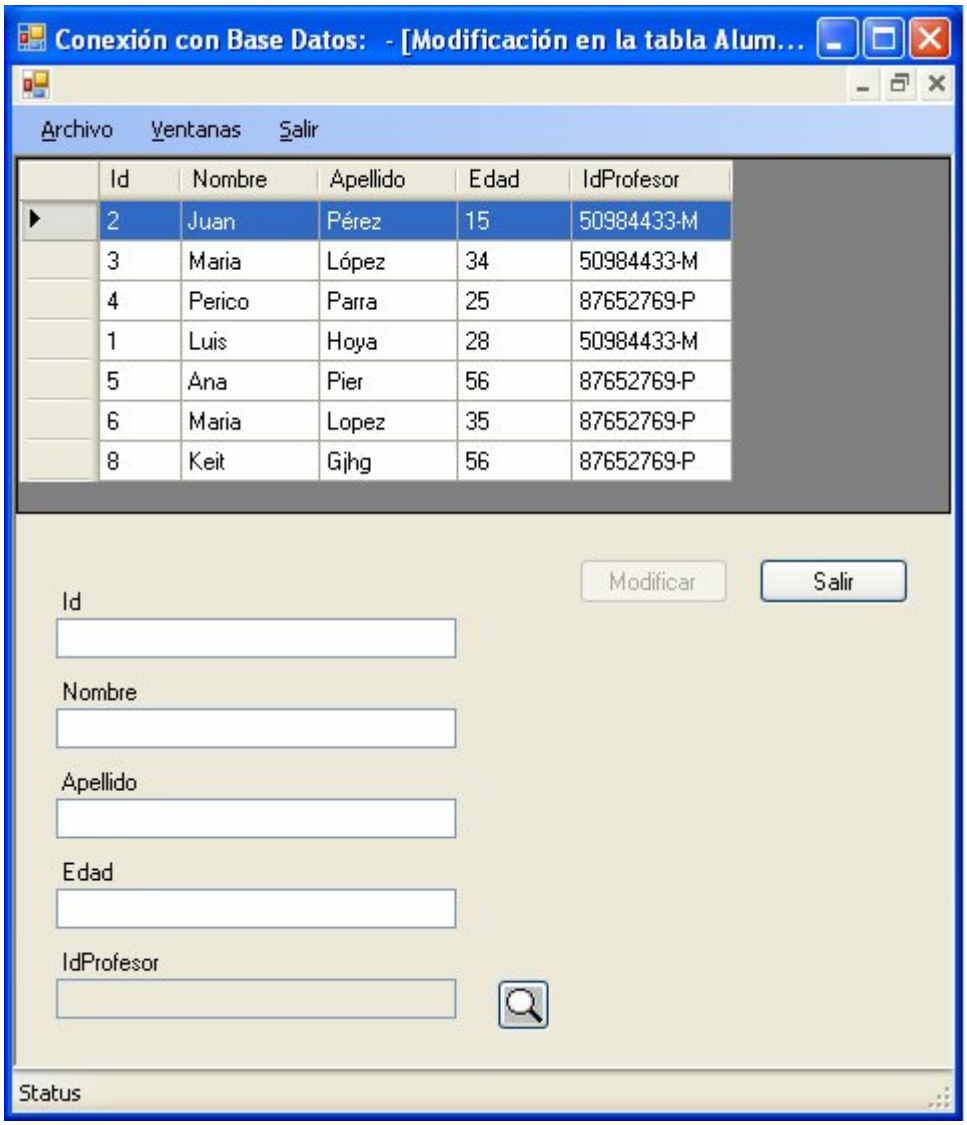
Pasos a seguir:

1. Se selecciona la fila del registro que deseamos eliminar.

2. Se pulsa sobre el botón "Eliminar" para solicitar la eliminación del mismo y se actualiza automáticamente la tabla que aparece en la parte superior del formulario.

Nota: con el botón “Salir” cerramos el formulario

- Modificación:



Pasos a seguir:

1. Se selecciona el registro a modificar en la tabla de la parte superior del formulario.

2. Una vez seleccionado los datos de la fila seleccionada se vuelcan cobre los campos del formulario desde donde son editados.

Nota: Los campos clave ajena no serán editables manualmente, sino a través de un control lupa que nos permitirá seleccionar la fila referenciada y se traerá a nuestro formulario origen la clave ajena correspondiente. Ver campo IdProfesor de la imagen anterior.

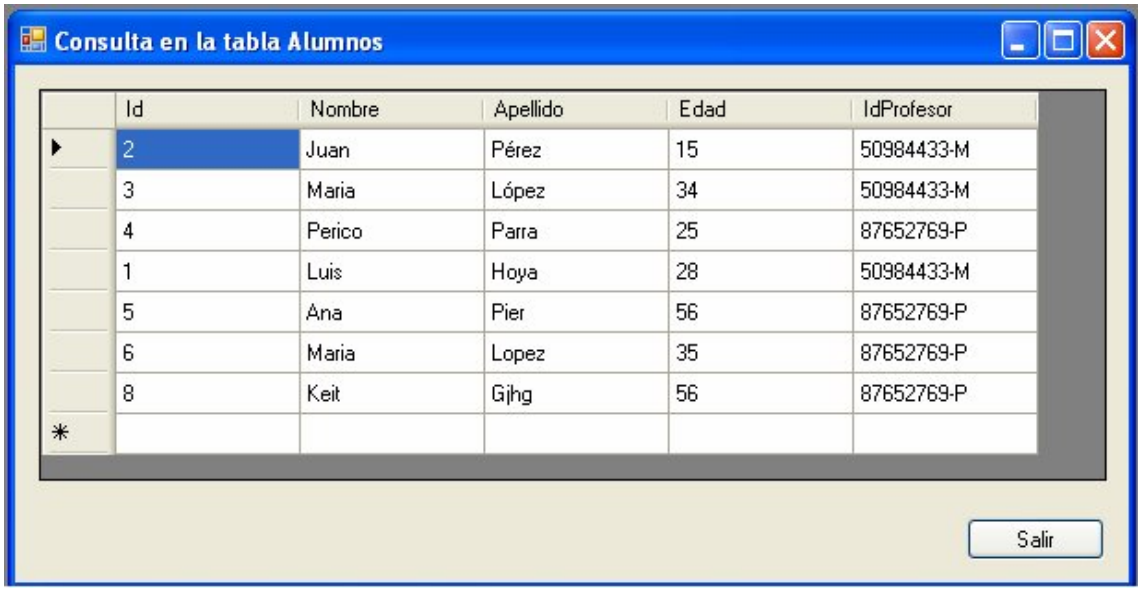
3. Se pulsa sobre el botón "Modificar" para solicitar la actualización del registro modificado y se actualiza automáticamente la tabla que aparece en la parte superior del formulario.

Nota: con el botón "Salir" cerramos el formulario

- Consulta:

En este formulario se muestran todos los registros de la tabla seleccionada.

Nota: con el botón "Salir" cerramos el formulario

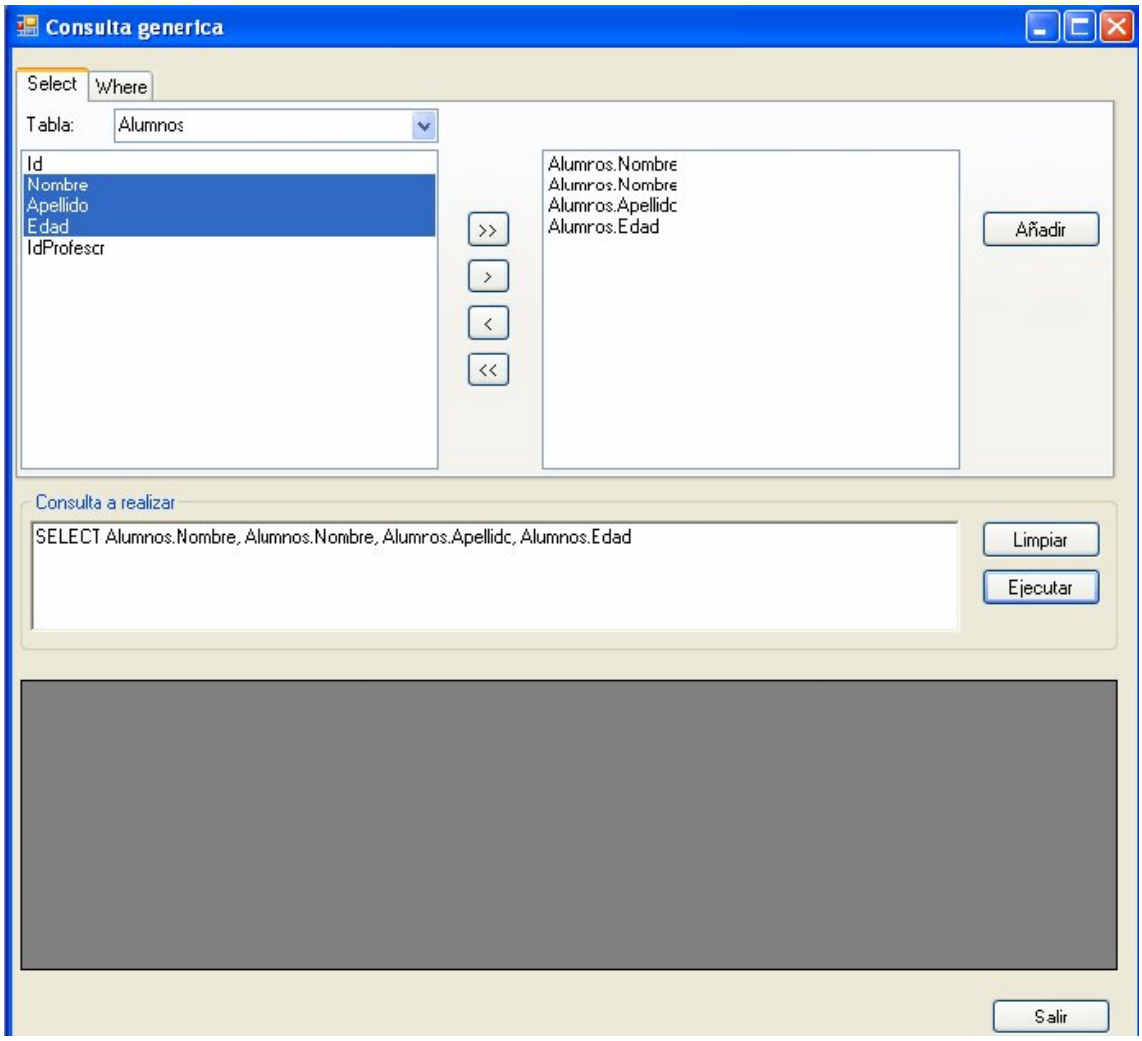


- Consulta Avanzada:

Esta consulta esta destinada a usuarios conocedores del lenguaje SQL.

Pasos a seguir:

1. Comenzamos creando la parte SELECT de la consulta:



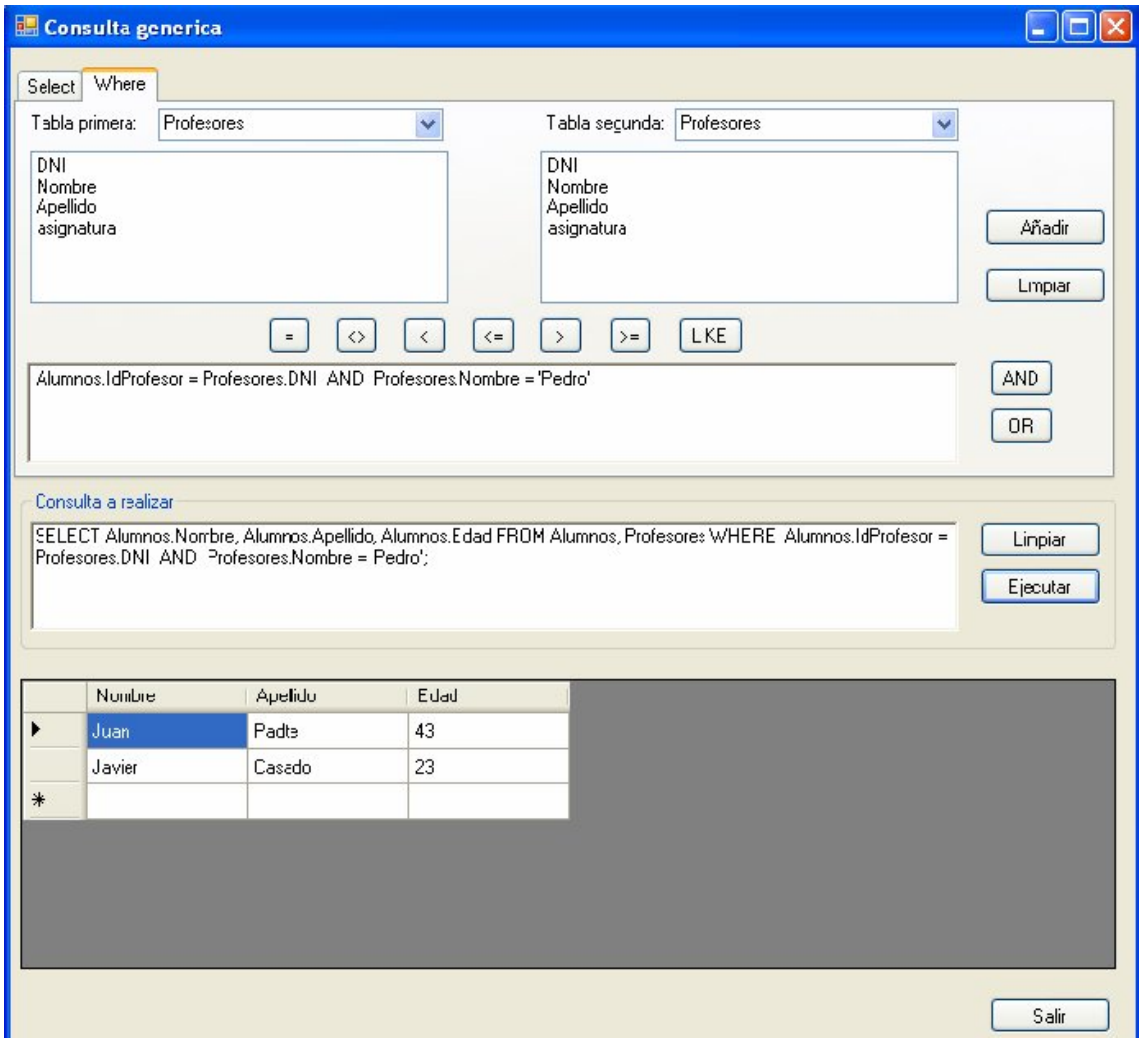
- Seleccionamos en la lista desplegable las tablas cuyos campos queremos que aparezcan en la parte select de nuestra consulta.
- Y se nos presentarán todos los campos de la tabla seleccionada en área de texto inferior.
- Desde este área de texto izquierdo, campo a campo vamos seleccionando los que queremos que aparezcan, ayudándonos de los botones:

- ">>" (añadir todos)
- ">" (añadir el seleccionado)
- "<<" (quitar todos)
- "<" (quitar el seleccionado)

- Una vez que tenemos en el área de texto derecho todos los campos deseados. Pulsamos el botón "Añadir" para construir la parte select de nuestra consulta, que aparecerá en el área de texto "consulta a realizar:".

  Esta área de texto puede ser limpiada con el botón "limpiar" o editada manualmente.

2. El siguiente paso es crear la parte WHERE, para ello:



- Seleccionamos en las listas desplegables las tablas cuyos campos queremos que aparezcan en nuestra cláusula where y se nos presentarán todos los campos de las tablas seleccionadas en las áreas de texto inferiores.

- Nos ayudaremos de los botones siguientes para crear la sentencia deseada:

    - Para comparar campos o campos con valores:

        - "=" (igual)

- “<>” (distinto)
- “<” (menor)
- “<=” (menor o igual)
- “>” (mayor)
- “>=” (mayor o igual)
- “LIKE” (como)

- Para crear conjunciones o disyunciones de varias sentencias:
  - “AND” (y)
  - “OR” (o)
- También podemos escribir directamente sobre el área de texto donde se va creado la cláusula where o limpiarla con el botón “Limpiar”

- Una vez tenemos la parte where construida la añadimos a nuestra consulta pulsando el botón “Añadir”. De la parte where.

3. Y ya en este punto, con la consulta a ejecutar construida, podremos:
   - Pulsar el botón “Limpiar” y borrar la consulta para crearla de nuevo.
   - Pulsar el botón “Ejecutar” y ejecutar la consulta creada. Cuyo resultado se mostrará a continuación según indicamos en la imagen superior

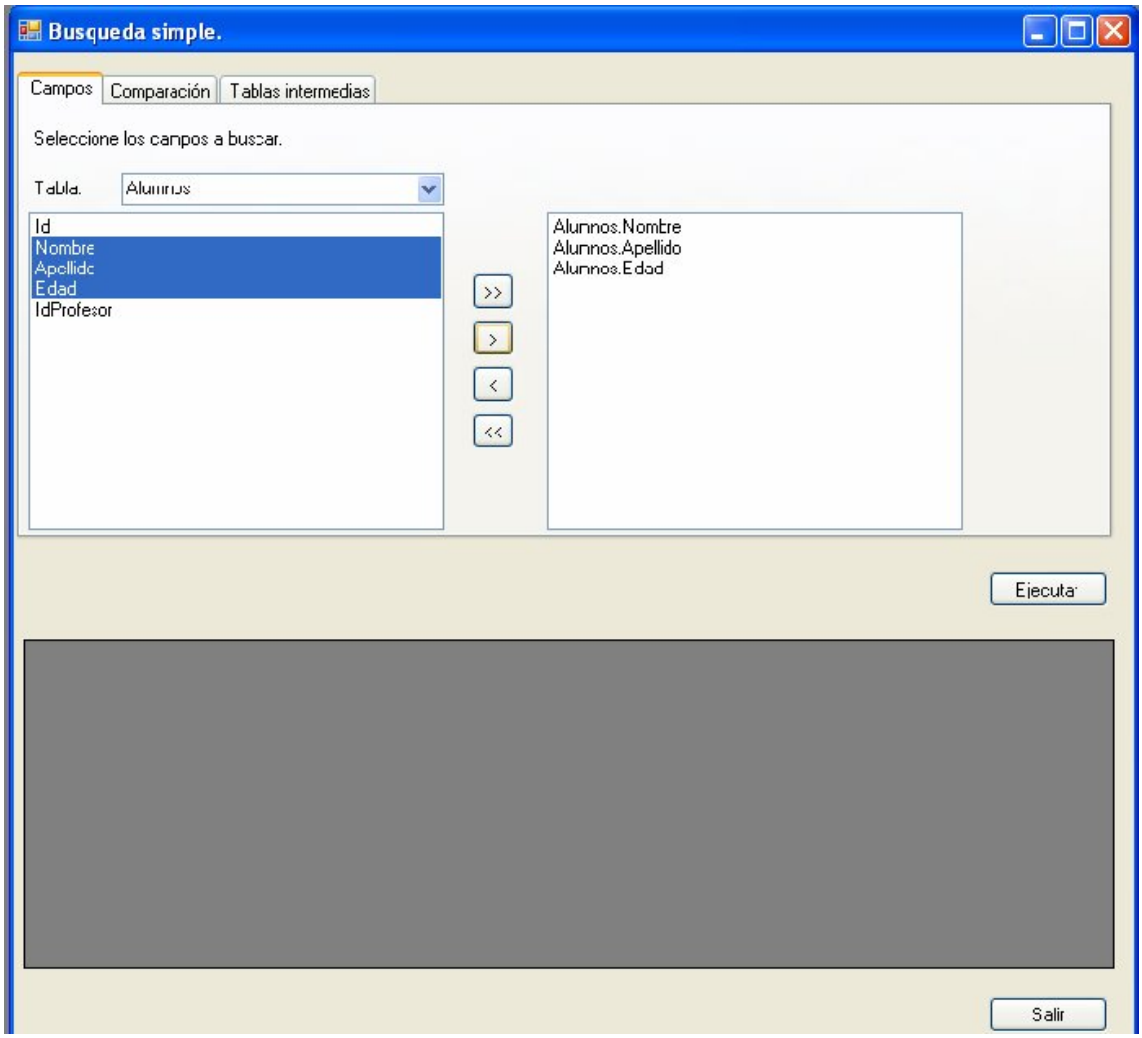
Nota: con el botón “Salir” cerramos el formulario

- Búsqueda:

Esta operación esta destinada a todo tipo de usuarios, ya sean conocedores o no del lenguaje SQL y permite hacer consultas sobre la base de datos.

Pasos a seguir:

1. Comenzamos seleccionando los campos de la búsqueda.



- Seleccionamos en la lista desplegable las tablas cuyos campos queremos buscar.

- Y se nos presentarán todos los campos de la tabla seleccionada en área de texto inferior.

- Desde esta área de texto izquierdo, campo a campo vamos seleccionando los que queremos que aparezcan, ayudándonos de los botones:

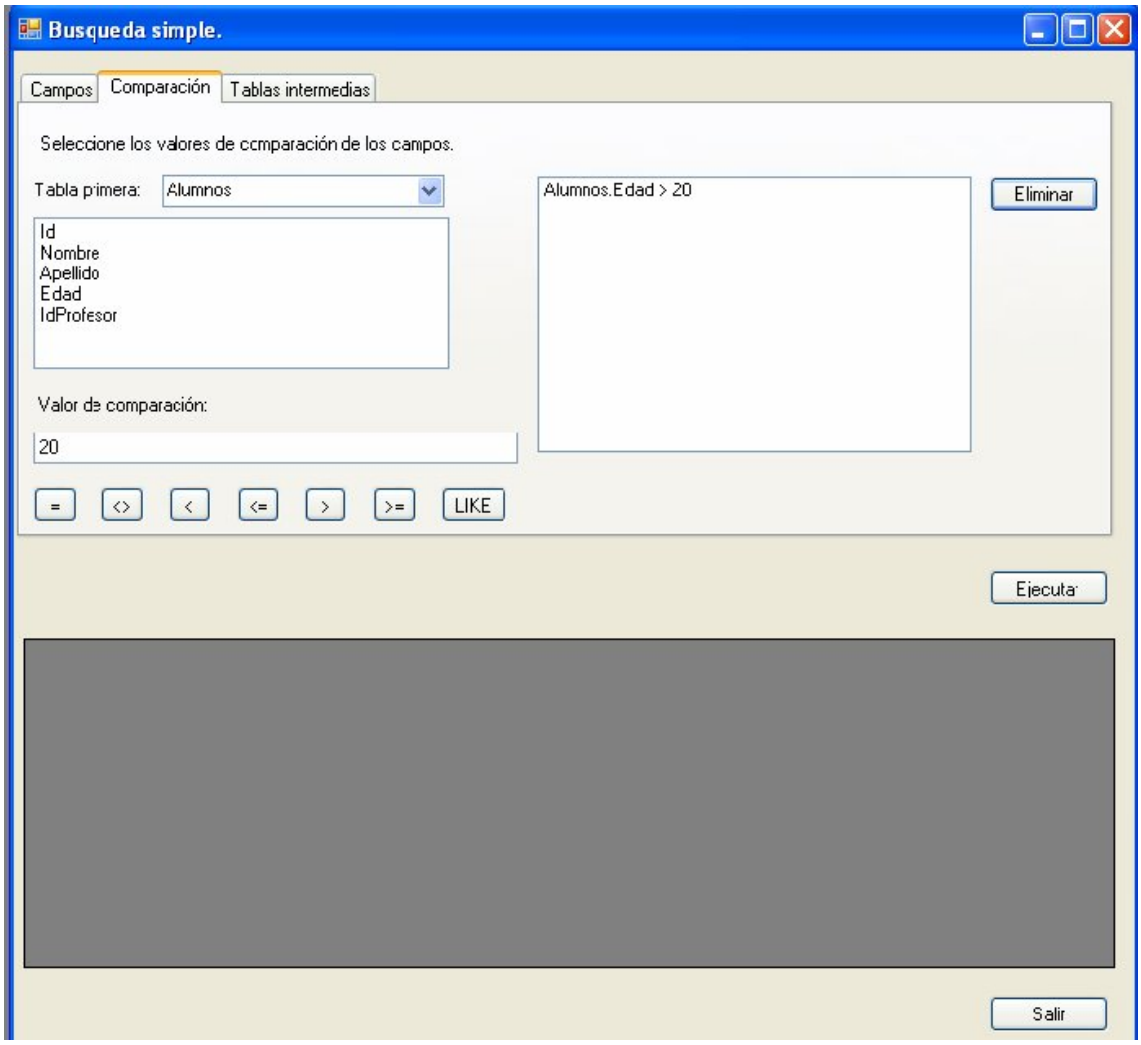
    - “>>” (añadir todos)

    - “>” (añadir el seleccionado)

    - “<<” (quitar todos)

    - “<” (quitar el seleccionado)

2. El siguiente crear las restricciones para la búsqueda para ello:



- Seleccionamos en la lista desplegable las tablas cuyos campos queremos que aparezcan en nuestra restricción de búsqueda. Y se nos presentarán todos los campos de la tabla seleccionada en el área de texto inferior

- Introducimos el valor de comparación con el que deseemos comparar el campo seleccionado en cada momento.

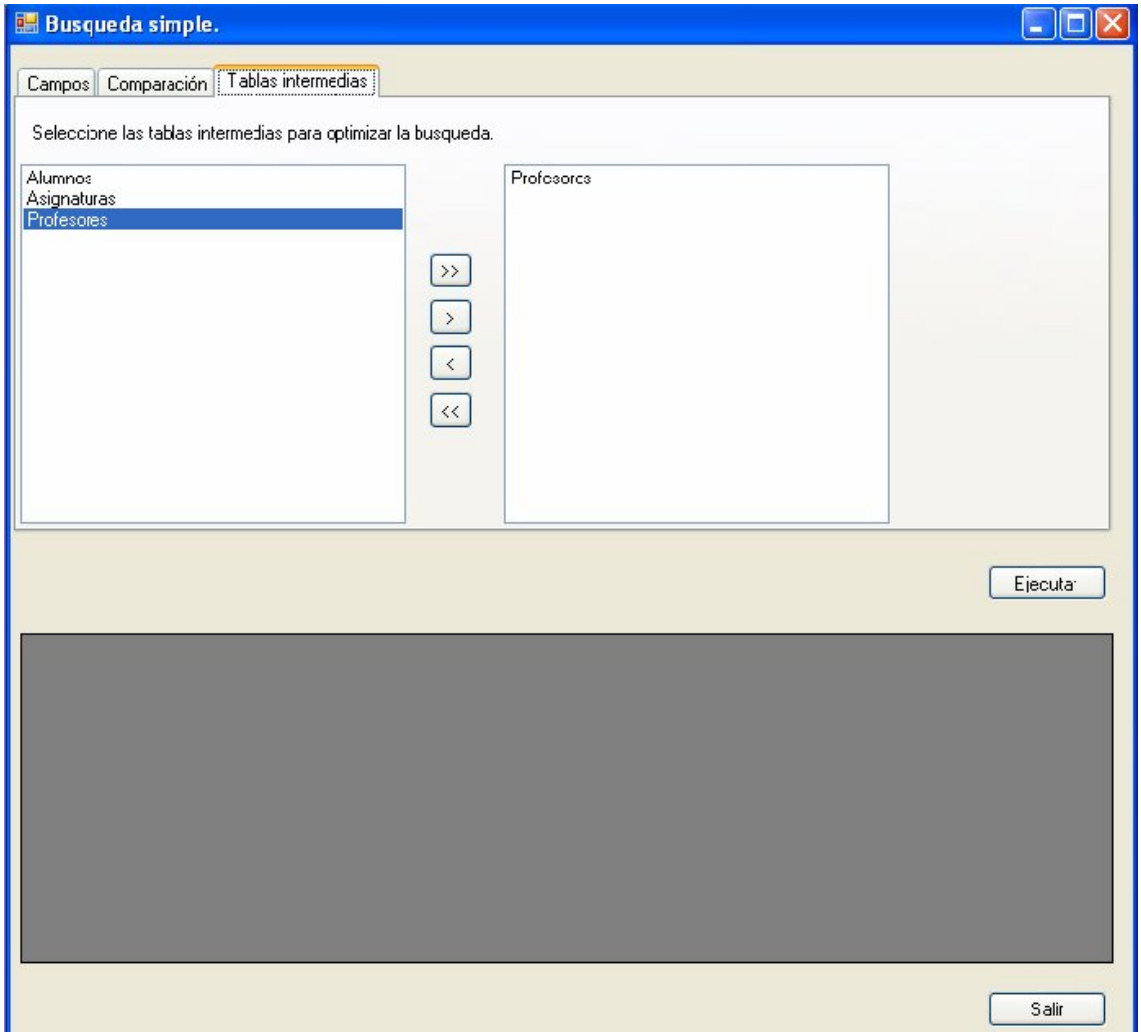
- Nos ayudaremos de los botones siguientes para crear la sentencia deseada:

    - Para comparar campos o campos con valores:

        - "=" (igual)

- “<>” (distinto)
- “<” (menor)
- “<=” (menor o igual)
- “>” (mayor)
- “>=” (mayor o igual)
- “LIKE” (como)

- Una vez tenemos creadas las restricciones pasamos a seleccionar las tablas intermedias que se deben atravesar para realizar la búsqueda deseada:



3. Y ya en este punto, con la búsqueda creada la ejecutamos. Pulsando el botón “Ejecutar”. Su resultado se mostrará al final de la página.

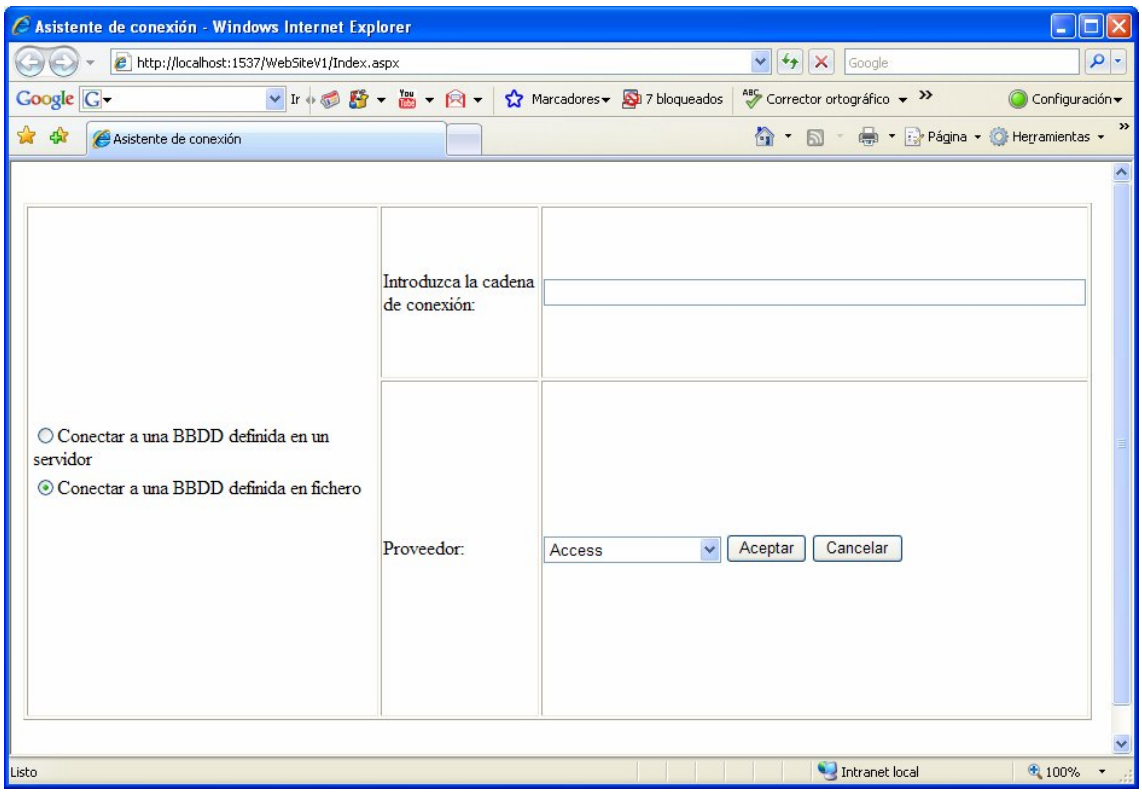
Nota: con el botón “Salir” cerramos el formulario

- Búsqueda Avanzada:

Esta operación es exactamente igual a la de Búsqueda simple explicada en las páginas anteriores, salvo que en este caso no es necesario indicar las tablas intermedias de la búsqueda a realizar.

Interfaz Web

Una vez instalado el programa no aparecerá la página principal del mismo:



El la que debemos realizar lo siguiente:

5. Elegir entre:

- Conectar a una base de datos definida en un servidor.
- O conectar a una base de datos definida en un fichero.

6. Introducir la cadena de conexión
7. Y seleccionar el proveedor deseado en la lista desplegable.
8. Aceptar o cancelar la operación

Si cancelamos la operación nos saldremos de la aplicación.

Si aceptamos existen dos posibilidades:

- En caso de haber seleccionado conectar a una base de datos definida en un servidor, tendremos que pasar por esta página previa, antes de llegar a la que comentaremos después.

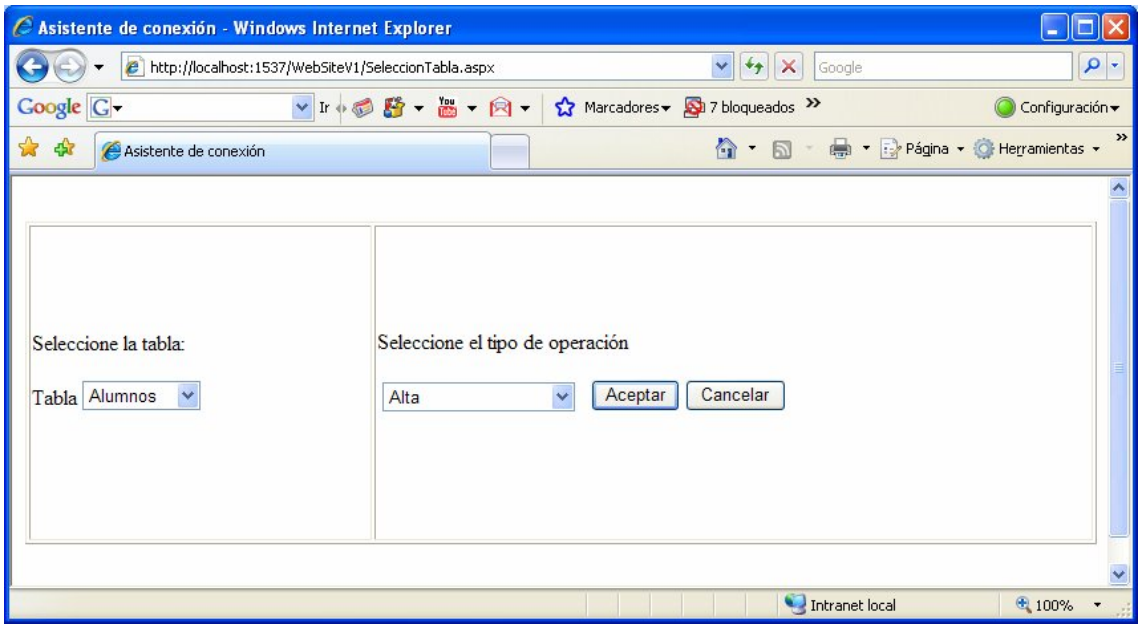
Pasos a seguir:

3. Seleccionar la base de datos concreta con la que queremos conectar, de entre las disponibles en el servidor.
4. Aceptar o cancelar la operación

Si cancelamos la operación pasamos a la página anterior, página principal del programa.

Si aceptamos la operación pasamos a la página que comentamos a continuación.

- En caso de haber seleccionado conectar a una base de datos definida en un fichero, pasaremos directamente (sin pasar por la página previa) a la página que mostramos a continuación:



Pasos a seguir:

Si lo que deseamos hacer es realizar un: alta, una baja una modificación o una consulta simple sobre una tabla concreta de la base de datos.

4. Seleccionamos, en la lista desplegable de tablas, la tabla de la base de datos sobre la que queremos operar.

5. Seleccionamos, en la lista desplegable de operaciones, la operación que queremos realizar sobre dicha tabla. Alta, baja, modificación o consulta simple.

En cambio si deseamos hacer una: consulta, una búsqueda o una búsqueda avanzada sobre la base de datos

2. únicamente seleccionaremos en la lista desplegable de operaciones, la operación que queremos realizar sobre la base de datos.

Una vez tomado el camino deseado el siguiente paso es aceptar o cancelar la operación.

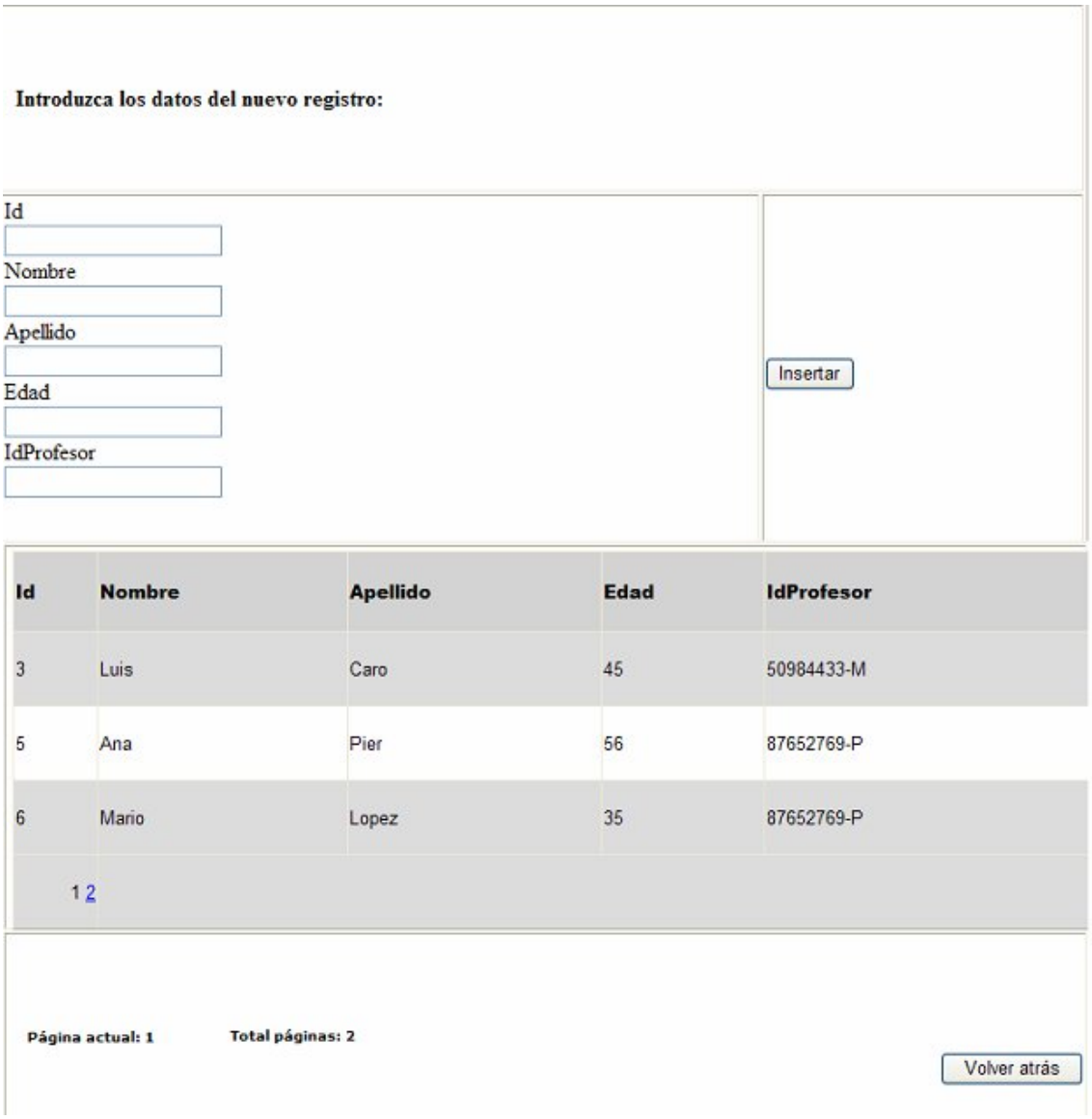
Si cancelamos la operación pasaremos a la página anterior que según por donde hayamos venido será:

- La pagina principal, en caso de haber seleccionado conectar con una base de datos definida en fichero
- La página de selección de base de datos, en caso de haber seleccionado conectar con una base de datos definida en servidor.

Si aceptamos la operación pasaremos a la página que se corresponda con la operación seleccionada.

A continuación explicaremos cada una de las páginas con las que operamos:

- Alta:



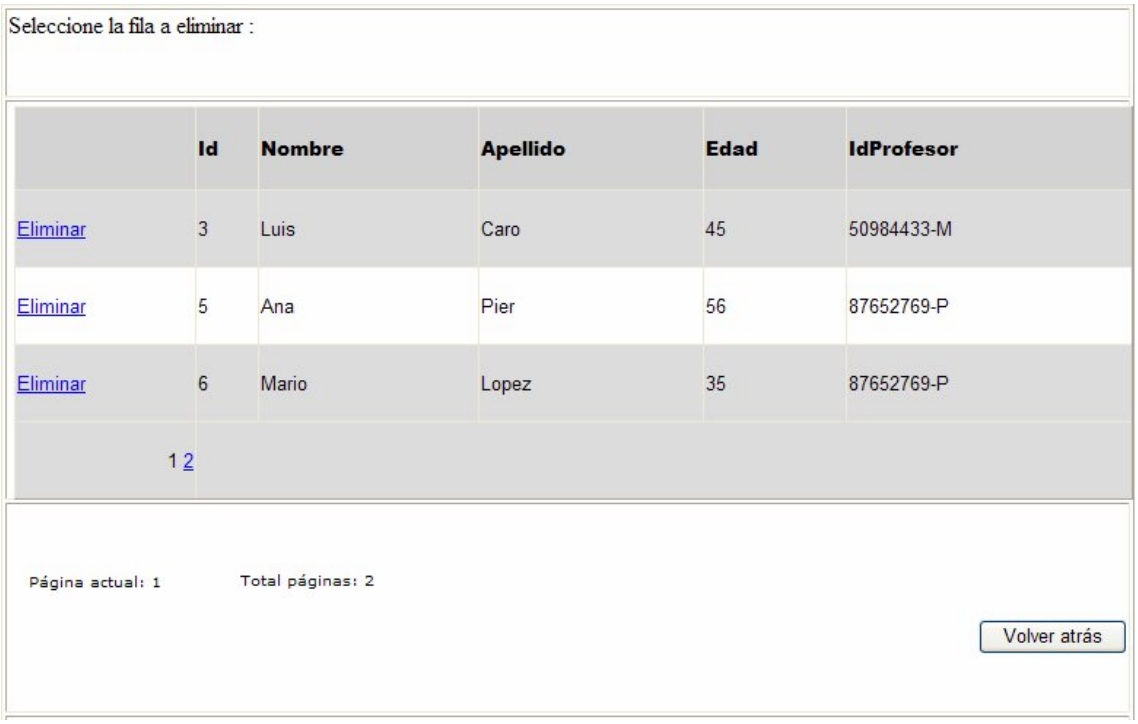
Pasos a seguir:

1. Se rellenan los todos los campos necesarios para el alta del nuevo registro.

2. Se solicita la inserción del registro en la tabla concreta. Actualizándose de manera automática los datos de la misma que se aparecen paginados en la parte inferior del formulario.

Nota: con el botón “volver atrás” o “Cancelar” volvemos a la página de selección de operación u operación y tabla. .

- Baja:

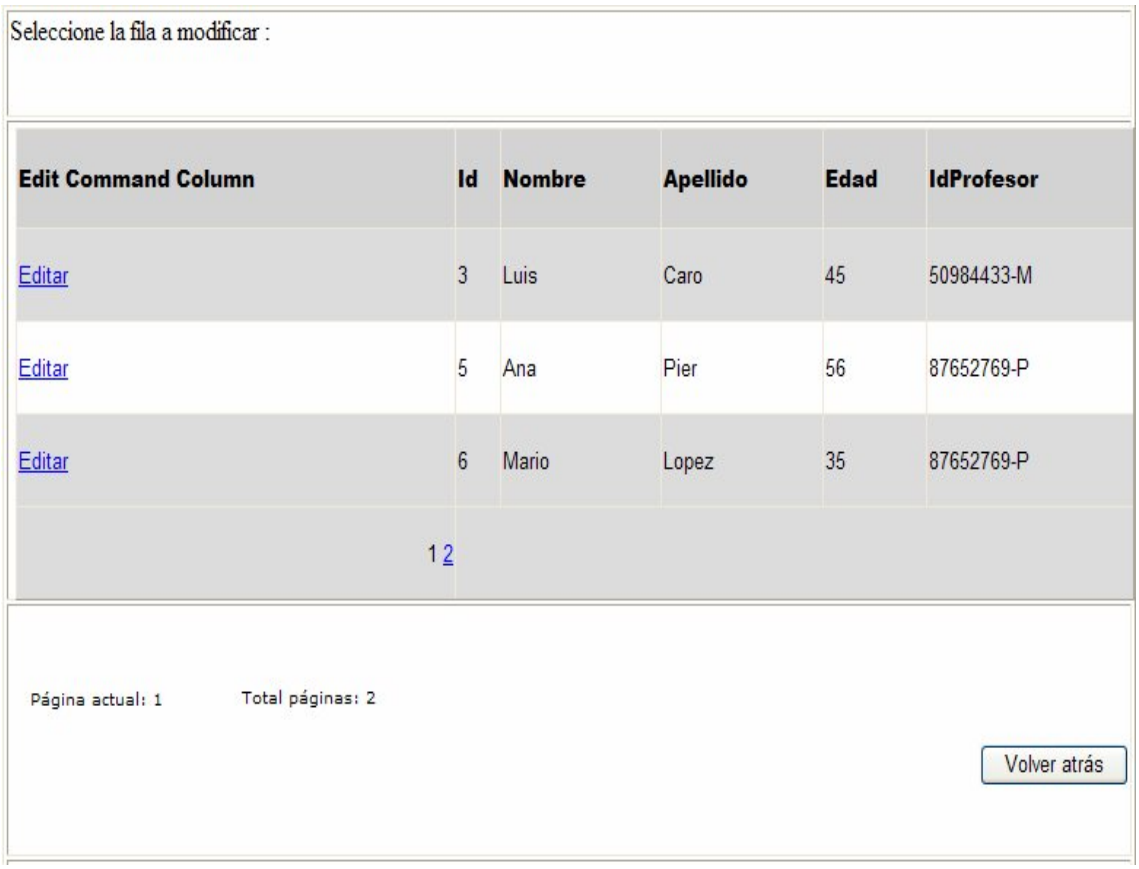


Pasos a seguir:

1. Se solicita la eliminación del registro pinchando sobre el link eliminar del mismo, Y se actualizan de manera automática los datos de la tabla a que aparecen paginados en la parte inferior del formulario.

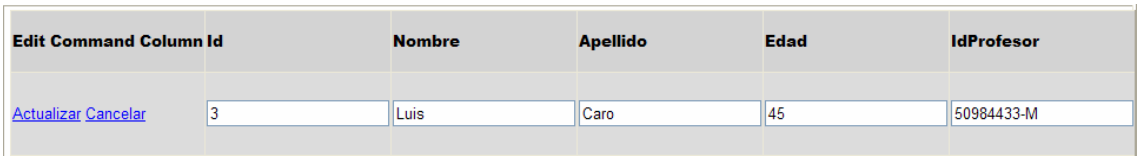
Nota: con el botón “volver atrás” “Cancelar” volvemos a la página de selección de operación u operación y tabla.

- Modificación:



Pasos a seguir:

1. Se edita un registro concreto de la tabla pinchando sobre el link editar del mismo.



2. Una vez editado el registro podemos:

   - Cancelar la operación solicitada, pinchando sobre el link cancelar del propio registro, de manera que el registro no se modifica.

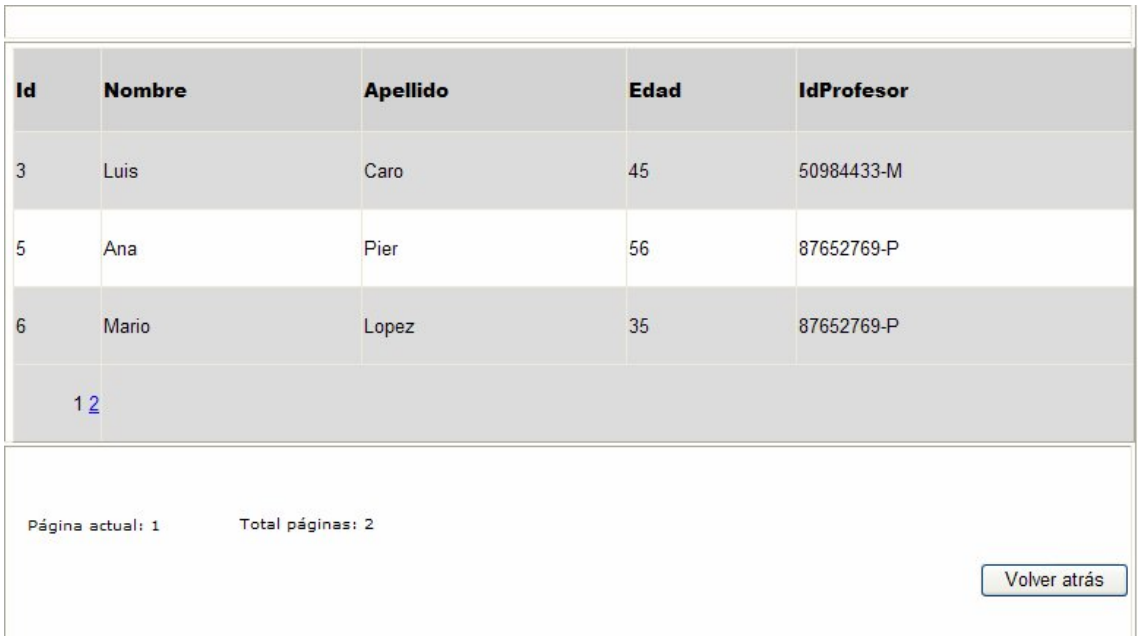
- Solicitar la actualización del mismo, pinchando sobre el link actualizar del propio registro, de manera que el registro queda actualizado.

Nota: con el botón “volver atrás” o “Cancelar” volvemos a la página de selección de operación u operación y tabla.

- Consulta simple:

En esta página se muestran todos los registros de la tabla seleccionada en la página previa:

Nota: con el botón “volver atrás” o “Cancelar” volvemos a la página de selección de operación u operación y tabla.

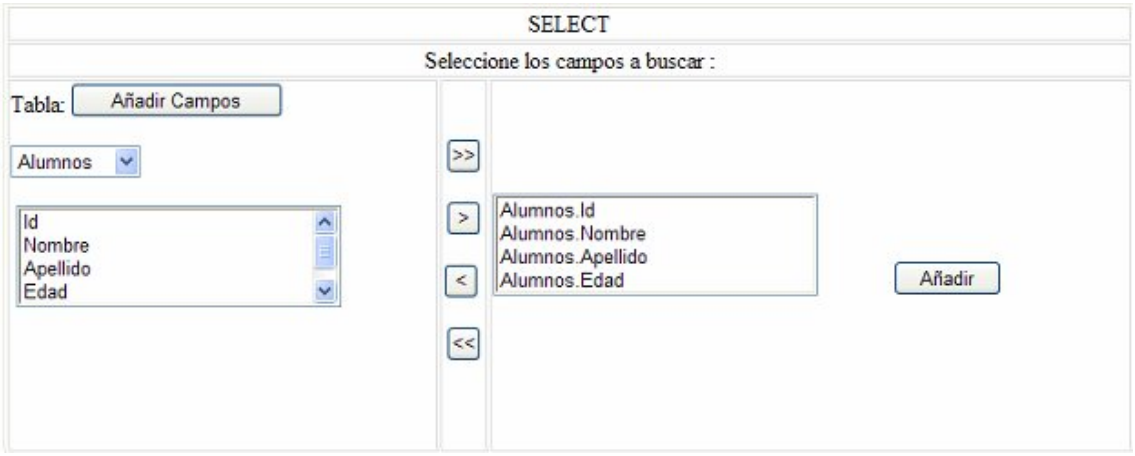


- Consulta:

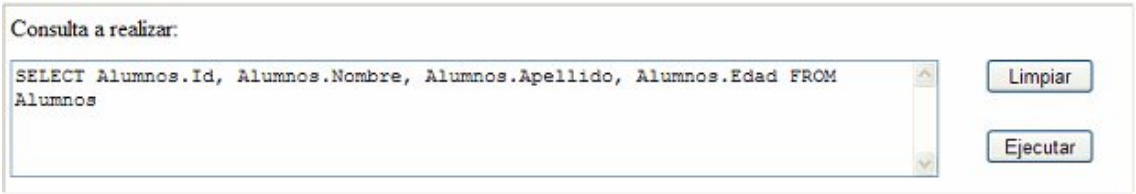
Esta consulta esta destinada a usuarios conocedores del lenguaje SQL.

Pasos a seguir:

1. Comenzamos creando la parte SELECT. de la consulta.



- Seleccionamos en la lista desplegable las tablas cuyos campos queremos que aparezcan en la parte select de nuestra consulta.
- Pulsando "Añadir Campos" y se nos presentarán todos los campos de la tabla seleccionada en área de texto inferior.
- Desde esta área de texto izquierdo, campo a campo vamos seleccionando los que queremos que aparezcan, ayudándonos de los botones:
    - ">>" (añadir todos)
    - ">" (añadir el seleccionado)
    - "<<" (quitar todos)
    - "<" (quitar el seleccionado)
- Una vez que tenemos en el área de texto derecho todos los campos deseados. Pulsamos el botón "Añadir" para construir la parte select de nuestra consulta, que aparecerá en el área de texto "consulta a realizar:"



2. El siguiente paso es crear la parte WHERE, para ello:

- Seleccionamos en las listas desplegables las tablas cuyos campos queremos que aparezcan en nuestra cláusula where. Pulsando "Añadir Campos" y se nos presentarán todos los campos de la tablas seleccionadas en las áreas de texto inferiores.

- Nos ayudaremos de los botones siguientes para crear la sentencia deseada:

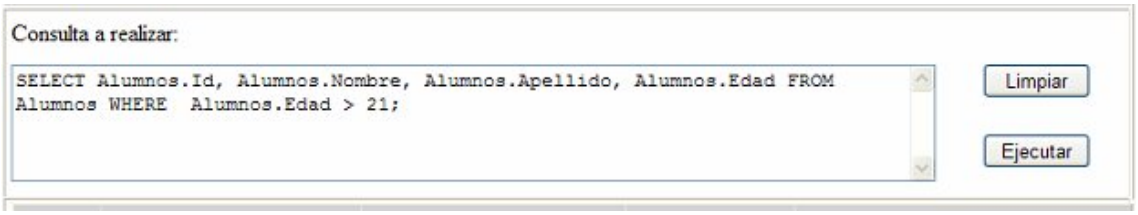
    - Para comparar campos o campos con valores:

        - "=" (igual)
        - "<>" (distinto)
        - "<" (menor)
        - "<=" (menor o igual)

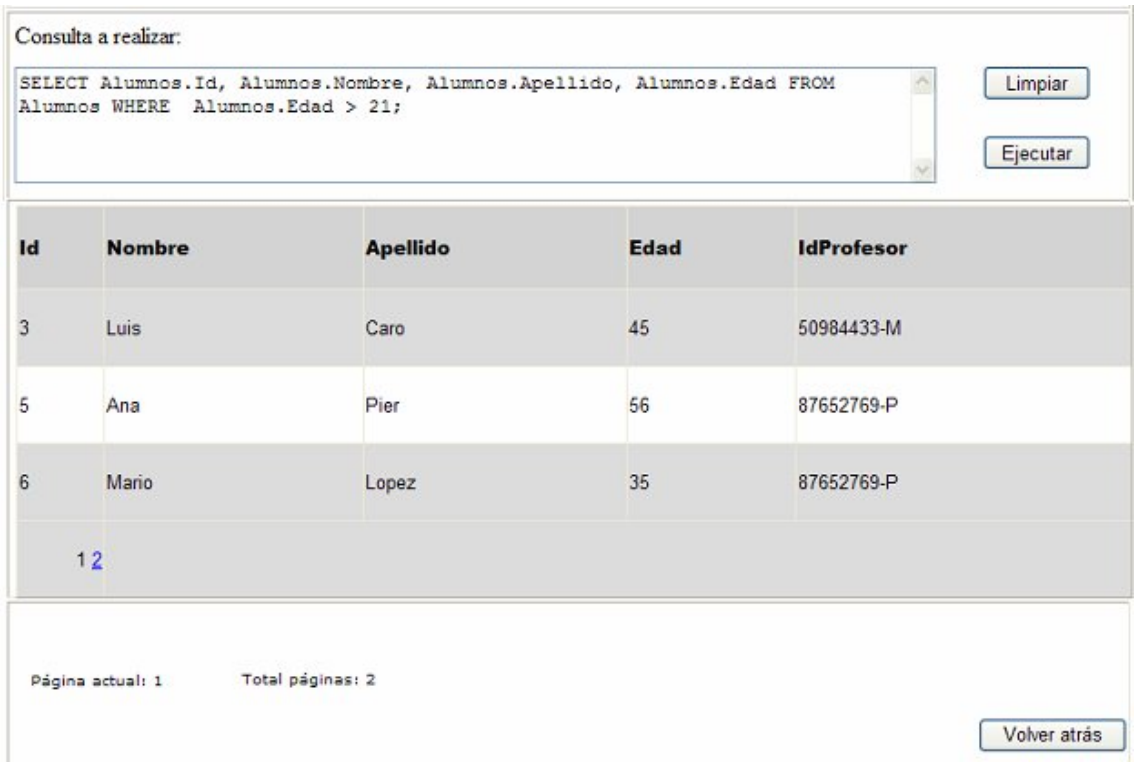
- “>” (mayor)
- “>=” (mayor o igual)
- “LIKE” (como)

- Para crear conjunciones o disyunciones de varias sentencias:
  - “AND” (y)
  - “OR” (o)
- También podemos escribir directamente sobre el área de texto donde se va creado la cláusula where o limpiarla con el botón “Limpiar”

- Una vez tenemos la parte where construida la añadimos a nuestra consulta pulsando el botón “Añadir”. De la parte where.



3. Y ya en este punto, con la consulta a ejecutar construida, podremos:
   - Pulsar el botón “Limpiar” y borrar la consulta para crearla de nuevo.
   - Pulsar el botón “Ejecutar” y ejecutar la consulta creada. Cuyo resultado se mostrará a continuación según indicamos en la siguiente imagen:



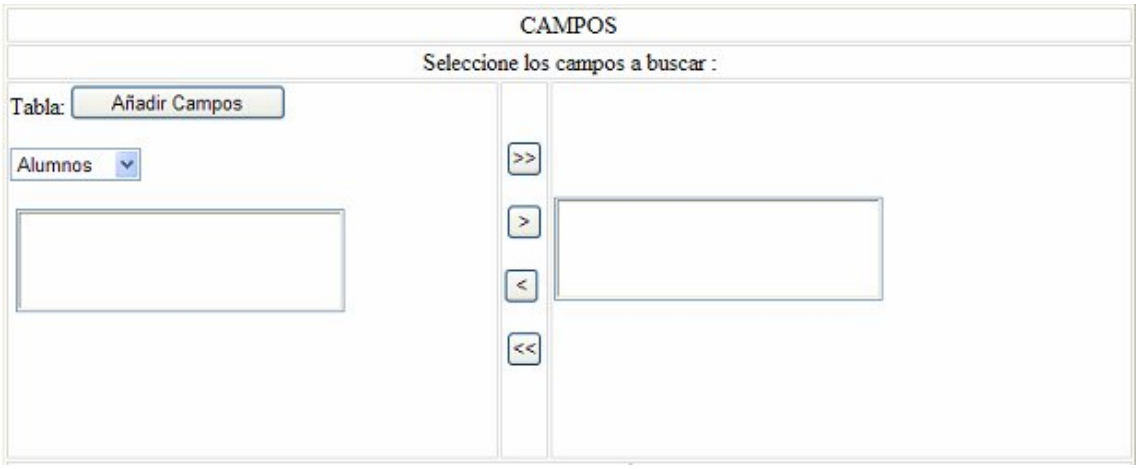
Nota: con el botón “volver atrás” o “Cancelar” volvemos a la página de selección de operación u operación y tabla.

- Búsqueda:

Esta operación esta destinada a todo tipo de usuarios, ya sean conocedores o no del lenguaje SQL y permite hacer consultas sobre la base de datos.

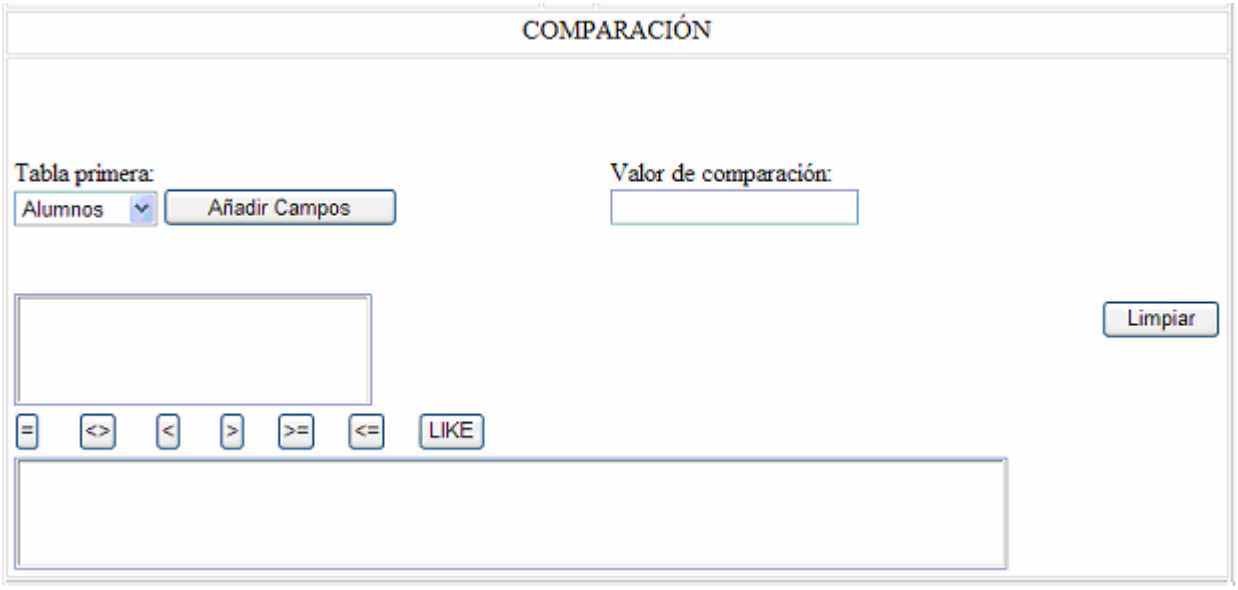
Pasos a seguir:

1. Comenzamos seleccionando los campos de la búsqueda.



- Seleccionamos en la lista desplegable las tablas cuyos campos queremos buscar.
- Pulsando “Añadir Campos” y se nos presentarán todos los campos de la tabla seleccionada en área de texto inferior.
- Desde esta área de texto izquierdo, campo a campo vamos seleccionando los que queremos que aparezcan, ayudándonos de los botones:
  - “>>” (añadir todos)
  - “>” (añadir el seleccionado)
  - “<<” (quitar todos)
  - “<” (quitar el seleccionado)

2. El siguiente crear las restricciones para la búsqueda para ello:

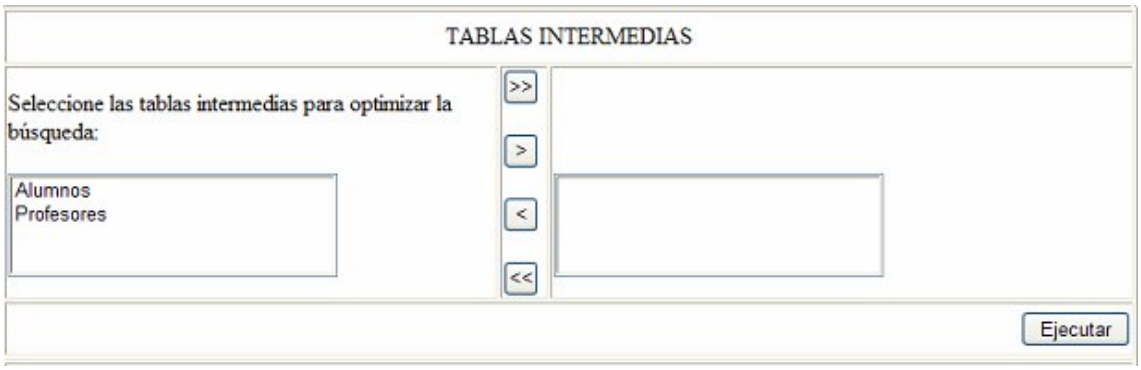


- Seleccionamos en la lista desplegable las tablas cuyos campos queremos que aparezcan en nuestra restricción de búsqueda. Pulsando "Añadir Campos" y se nos presentarán todos los campos de la tabla seleccionada en el área de texto inferior
- Introducimos el valor de comparación con el que deseemos comparar el campo seleccionado en cada mioemento.
- Nos ayudaremos de los botones siguientes para crear la sentencia deseada:
    - Para comparar campos o campos con valores:
        - "=" (igual)
        - "<>" (distinto)
        - "<" (menor)
        - "<=" (menor o igual)
        - ">" (mayor)

- “>=” (mayor o igual)
- “LIKE” (como)

- También podemos eliminar restricciones pinchando sobre la restricción a eliminar y pulsando el botón “Limpiar”.

- Una vez tenemos creadas las restricciones pasamos a seleccionar las tablas intermedias que se deben atravesar para realizar la búsqueda deseada:

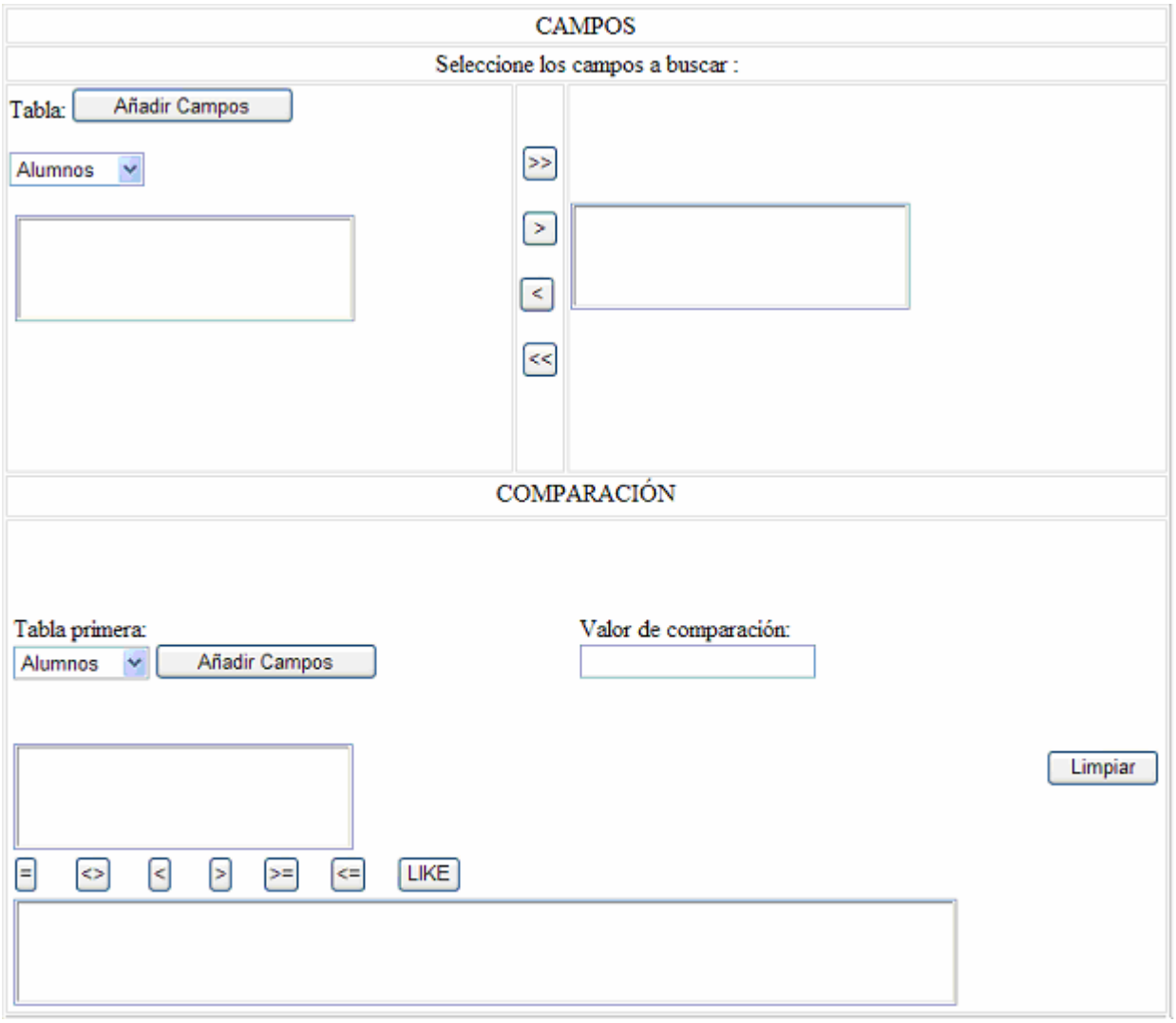


3. Y ya en este punto, con la búsqueda creada la ejecutamos. Pulsando el botón “Ejecutar”. Su resultado se mostrará al final de la página.

Nota: con el botón “volver atrás” o “Cancelar” volvemos a la página de selección de operación u operación y tabla.

- Búsqueda Avanzada:

Esta operación es exactamente igual a la de Búsqueda simple explicada en las páginas anteriores, salvo que en este caso no es necesario indicar las tablas intermedias de la búsqueda a realizar.


CAMPOS
Seleccione los campos a buscar :
Tabla:
Añadir Campos
Alumnos
>>
>
<
<<
COMPARACIÓN
Tabla primera:
Alumnos
Añadir Campos
Valor de comparación:
Limpiar
=
<>
<
>
>=
<=
LIKE

Autorización de difusión

Autorizamos a la Universidad Complutense a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, nosotros, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado.

Fdo.

Irene Parrón Carrascal -DNI: 50982211C

Javier González Chacón -DNI: 52881629Y

Jose María Marín del Valle -DNII: 46886048L